Modelling Sheep Flock Interaction with Sheepdogs

Ben Buckley

Bachelor of Science in Computer Science with Honours
The University of Bath
April 2008

Modelling Sheep Flock Interaction with Sheepdogs

Submitted by: Ben Buckley

Abstract

Flocking is a natural phenomenon which has been modelled on computers for many years. This project looks at the existing modelling techniques and attempts to adapt and enhance them in order to produce a realistic Agent Based Model (ABM) of sheep flocks. We then extend the model to include the modified predatory behaviour of sheepdogs to investigate the interaction between sheepdogs and sheep flocks. Multiple sheepdogs are modelled, using behaviour similar to that of a group of lions hunting cooperatively. Finally, an extension to the ABM is introduced, based on the emotion and personality of sheep and sheepdogs. The project is approached iteratively, so that emergent behaviour can be taken into account at various stages, and considered in subsequent phases of the design. Results show that the sheepdog's task can be made easier or more difficult depending on the personalities of the sheep which make up the flock, herding a flock is shown to be more problematic when it contains a dominant sheep. Different steering techniques and formations of sheepdogs are shown to have varying levels of effectiveness, both in terms of speed and control of the flock. In comparison to a single sheepdog, multiple sheepdogs are shown to be faster at gathering sheep into flocks and are shown to have greater control when herding a flock.

Table of Contents

Figure Index

Acknowledgements

I would like to thank Claire Willis for her help and advice throughout the project. I would also like to thank James, Katherine and Liam for helping to proof-read various sections of this document.

Chapter 1, Introduction

Sheep, or *Ovis Aries,* are flocking animals by nature. Despite popular belief, they have been observed to be just as intelligent as cattle, and not far behind the widely noted IQ of pigs [Weaver, 2005]. An interesting example of this intelligence was exhibited by crafty sheep in Yorkshire, who discovered a means of crossing cattle grids by rolling on their backs, in order to raid the fresh flora of local gardens [BBC News, 2004].

The natural desire to flock, combined with the tendency to follow a more dominant counterpart, made sheep one of the first animals to be domesticated for farming [Budiansky, 1999]. Shepherds are able to use sheepdogs to take advantage of this behaviour in order to move and control their flocks, although the various breeds and personalities of sheep can make this task very difficult [Downe, 2005]. In general, flocking behaviour is common among many prey animals as a form of defence against predators [Gueron et al., 1996] . The ability to move as what appears to be a single, centrally controlled entity helps to confuse predators, making it difficult for them to focus on a specific target. This supposed group safety allows animals to forage without having to constantly look out for predators.

The overall term for modelling living creatures on a computer is known as Artificial Life or Alife [Dictionary.com, "Artificial Life" 2008]. The challenge of developing a flocking simulation on a computer is difficult since it aims to produce what looks like a complex formation of individuals, when in actual fact there is no clearly defined formation pattern or thought process which the flocking animals go through [Lindhé, 2004] . It was once thought that the formation of a flock was achieved through a complex series of interactions between living organisms [Gabbai, 2005]. This idea was unchanged until Reynolds [1987] proposed an idea that flocking is simply the emergent behaviour produced by a series of simple local rules. The local rules were defined as follows, where each animal must consider the following:

- Alignment – Ensuring that it is facing, on average, the same direction as its nearest neighbours.
- Cohesion – Ensuring that it is close to the centre of the flock, with regards to its nearest neighbours.
- Separation – Ensuring that it is not too close to those around it, avoiding collisions.

Reynolds [1987] applied these rules to birds, creating the famous *Boids* simulation. With simple alterations to the weighting of each rule, Reynolds was also able to model fish and insects. To model sheep, the idea is applied to a 2D environment, rather than 3D. A 2D flock is more commonly referred to as a herd.

To 'shepherd' something means to move or guide it in a particular direction, whereas the term 'shepherd' traditionally refers to a person employed to guard and tend to sheep. In this project, the term will be used to represent any autonomous being which influences others by exerting repulsive forces on them, potentially taking advantage of their flocking behaviour.

Aside from modelling the interaction between sheep and sheepdogs, a shepherding simulation could be applied to a variety of situations. One of the most practical applications to date involves a robotic shepherd, used to herd live animals [Vaughan et al., 1998]. As the behaviour of flocking animals is understood in more detail, it will be possible to design and engineer more sophisticated autonomous shepherds, potentially providing cheaper, robotic alternatives to sheepdogs. Another application of flocking behaviour has been used to create animations of realistic flocks for use in films such as *The Lion King* [Gabbai, 2005]. Therefore, the potential lies for a realistic shepherding model to be applied to both the film and computer game industries.

The most well known types of shepherding are those of gathering and herding, where the shepherd collects individuals into a flock and moves the flock as a whole to a particular destination. While it will be useful to model these behaviours in terms of sheep and sheepdogs, there are also more generic applications such as modelling how predators work together to hunt, or 'gather' their prey or developing robots that can gather up oil from an oil spill. A more creative application, as far reaching as into the area of medical research, is modelling how neurons migrate through the brain [Lien et al., 2004]. A variety of other applications have been identified, one form of shepherding, known as covering, involves staying nearby to a flock as it moves along a given path. Developing an autonomous robot to perform such a task could be used for mine sweeping. A further variation on the theme of shepherding is that of patrolling, that is, keeping a flock away from a particular area.

Overall, there is a great deal of work to be done in order to understand how shepherds can control animals which have a natural flocking instinct. This project aims to develop a simulation of the flocking behaviour exhibited by sheep, and the shepherding behaviour displayed by sheepdogs. It is feasible that the techniques used to model the sheepdogs could be applied to a more generic situation, as described above.

The initial challenge in this project will be to adapt and extend Reynolds' [1987] original flocking rules to create a realistic sheep flocking model. Once a suitable way of representing flocks has been implemented, the action selection mechanisms and behaviours of the sheepdogs will be developed. This involves developing a method of creating spontaneous coordination between sheepdogs so that multiple sheepdogs can work together constructively. On top of all this, we will investigate how to model different personalities of sheep, related to the different character traits of particular breeds such as lowland sheep [Downe, 2005]. This will include looking at how emotion can affect the outcome of an agent based simulation, investigating emotions such as fear and frustration in both sheep and sheepdogs.

1.1 Project Layout

Following this section, the literature review (**Chapter 2**) will concentrate on work done so far in this area, and on researching a suitable amount of background detail on sheep and sheepdogs, allowing us to identify which research is useful and where we can provide a contribution to the field.

Using the knowledge gained from the literature review and the original aims of this project, a series of research objectives will be defined (**Chapter 3**). These objectives will form the basis of the design of the model and will give a marker against which to assess the success of the project.

The design section of this project was approached iteratively. Initially, in **Chapter 4**, some of the more general, higher level design decisions are considered. The sheep model is designed in **Chapter 5**, and is put through several stages of design and testing. The various improvements made at each stage are discussed. **Chapter 6** focuses on the design of the sheepdog model, including iterations for the different types of sheepdog. Following the design sections there is a brief overview of a selection of the interesting implementation details for the entire project (**Chapter 7**), this includes a class diagram and a high level description of how the simulation is structured.

Although each part of the model was tested and evaluated as development progressed, only the final, integrated system was considered and evaluated in testing and results (**Chapter 8**). This includes observations and a discussion of the simulation as well as a series of tests which are run to see how the interactions between sheep and sheepdogs actually work. Once a suitable model has been developed, it will be employed to analyse the interactions of the sheep and sheepdogs with various parameters implemented. These results, also included in this chapter, will allow us to discuss to what extent we have met the original research objectives. Finally, using the evaluation of these results, along with the general observations made throughout the project, an overall conclusion is drawn (**Chapter 9**), including a critical analysis of the project, potential future work and a summary of the project as a whole.

Chapter 2, Literature Review

2.1 Introduction

Within this project there were several different areas that needed to be investigated in order to gain a full understanding of what can be done in terms of modelling flock behaviour and interaction with sheepdogs, while taking into account individual animals' personalities, emotions and moods.

It was important to have at least a basic understanding of the various types of animals that might be involved in such situations, and in particular for this project, sheep and sheepdogs. It was also useful to look elsewhere in nature to find analogous systems and situations such as the behaviour of pack animals when hunting.

A well documented area of computer science is the modelling of flocks, we explore the work done so far and what techniques have been used, and identified which techniques can be extended and improved.

In terms of modelling interactions between sheepdogs and sheep it was useful to investigate predatory behaviours in general, looking at how predators may work together when attacking a group of prey. It was also useful to look at how flocks react to such predatory attacks and the mechanisms that might be employed to defend against or avoid them.

A final, but no less significant area, was to look at what has been done in terms of modelling emotions, moods and personalities in multi-agent systems. This involved looking at how emotion can affect both the behaviour of sheep and sheepdogs as well as identifying a suitable method for modelling their emotions.

Overall this section sets the scene for the project, providing a solid background while also posing interesting questions to be answered or experiments to be carried out. The aim was to find some areas where there is clearly room for improvement or further research, in order to give this project a unique angle on the various problems and ideas involved with computer based simulations.

2.2 Background Research

In order to create a realistic simulation, it is necessary to have a solid understanding of the animals whose behaviour we are attempting to simulate. The following section looks at various sources, ranging from papers to websites containing "word of mouth" information, since there are few formal papers on sheep herding. Because of this, it will also be useful to look at interactions between similar animals and their prey, such as the predatory behaviour of lions, wolves and African hunting dogs.

It is also useful to have a more explicit definition of what is meant by flocking. McFarland gives a definition for flocking in birds, describing it as a group of birds which remains together due to social attraction, he contrasts it to an aggregation which is simply a collection of individuals reacting in the same way to some external event in the environment [McFarland, 2006 ("Flocking in Birds")]. Birds which are flocking will show some form of coordinated behaviour such as the direction and speed at which they are flying. There is a mutual benefit in that it dilutes the effect of predator attacks and increases overall vigilance within the flock. It is these mutually beneficial properties which separate flocking behaviours from self-interested behaviours.

2.2.1 Sheep

It is difficult to find officially published papers in any discipline that give a deep insight into sheep and sheepdog behaviours and characteristics. However there is a wealth of knowledge spread over sheep herding websites, hobbyist websites and general online encyclopaedias relating to sheep and sheepdogs. This project aims to examine the interaction between flocks in general and with one or more sheepdogs, therefore it is not so important to conduct extensive research on sheep and dogs. Due to the lack of published information on sheep in general, it was also beneficial to look at analogous animals, such as looking at the flocking behaviours of birds, or the predator-prey interactions between lions and gazelles.

In land animals such as sheep, the cooperative movement behaviour is often referred to as herding rather than flocking. The main benefit of herding to an individual is the increased chance of detecting a predator, since there are many animals all looking out for the same threat. This is especially beneficial to grazing animals who cannot both graze and watch for predators at the same time, a herd allows some animals to graze while others stand guard, looking out for predators [McFarland, 2006 ("Herding")]. An individual has a lower probability of being attacked by a predator when it joins a herd- the larger the herd, the lower the probability.

It should be noted that sheep and other flocking animals will not flock all of the time. Without the threat of a predator, sheep have a much desire to flock [Downe, 2005]. It is claimed that sheep only exhibit flocking behaviour when in the presence of at least four other sheep [Weaver, 2005]. More importantly, the need to graze affects how much flocking occurs, since grazing is inevitably a more successful task when the herd is spread out over a larger area. Therefore, the behaviours of sheep can be seen as striking a balance somewhere between having enough space to allow for sufficient grazing, and being close enough to others to satisfy the desire to be with the flock [Delgado-Mata and Ibanez-Martinez, 2006]. The defensive behaviour that sheep exhibit is sometimes referred to as the *prey drive* [Prey

Drive, Wikipedia], this is the instinct to either run away from, or, stand up to a predator and attempt to defend itself. This 'Fight or Flight' decision [Fight or Flight, Wikipedia] can be influenced by the sheep's personality, emotions and mood. It is unclear whether or not this prey drive is actually coordinated behaviour, aimed at benefiting the flock as a whole through each individuals' actions, or whether it simply emerges from the aggregate anti-predator reactions of each individual. A sheep will react to a predator when it enters its *flight zone* [Flight Zone, Wikipedia] [Downe, 2005], this will generally cause sheep to flee but in certain situations, such as a sheep being cornered, the sheep may ram or bite predators in an attempt to escape [Weaver, 2005].

Comparing McFarland's [2006] definition of flocking to herding raises the question as to where the flocking behaviour ends for sheep and where aggregate behaviour begins. Sheep certainly do exhibit some flocking behaviour, as mentioned above, to improve predator defence and aid grazing efficiency, however, reactions to predator attacks may be largely thought of as an aggregate behaviour, rather than some coordinated defensive manoeuvre.

Various different personalities of sheep have been identified, particularly the *Outlier* and the *Bellwether* [Sheep, Animal Corner, 2008]. An *Outlier* is a sheep that has a tendency to wander away from the flock to graze elsewhere, usually because it is too weak to graze successfully in close quarters with other sheep. The strongest sheep will often fight their way to the centre of the flock so that they can graze without having to worry about detecting predators. Another personality of sheep is that of the *Bellwether* which is a sheep that will naturally lead the flock, this is usually a castrated ram.

In situations where there are no natural predators it has been observed that sheep will not exhibit strong flocking behaviour [Smith et al., 1997]. For the purpose of this project it is more beneficial to assume that the interaction with sheepdogs counts as a natural predator, therefore the sheep will still exhibit a strong flocking behaviour. When a sheep becomes separated from the flock it is likely to become stressed [Simmons and Ekarius, 2001], potentially making it more sensitive to predators and less predictable to sheepdogs if it panics. Different breeds of sheep exhibit different characteristics of flocking, for example the *Merinos* sheep form a very tight flock and will not usually form a sub-group unless there is a severe food shortage. However, breeds such as *Southdowns* and *Dorset Horns* will often form sub-groups [Sheep, Animal Corner, 2008]. Differences have also been observed in the reactions of different breeds to predators, sheep reared on the hills are known as *light sheep* and are more likely to flee from predators at a further range than other breeds. Conversely, sheep such as *Southdowns* which descend from lowland sheep have smaller flight zones and are more reluctant to flee from predators, forming tight cohesive flocks, these are known as *heavy sheep* [Downe, 2005]. These apparent differences in behaviour between breeds could be attributed to certain sheep having different personality traits, which will be interesting when we look at modelling emotion, mood and personality in an agent based simulation.

2.2.2 Sheepdogs

There is little literature available from published sources that gives a useful background to sheepdogs. The most respectable source is from an encyclopaedia found on the New Zealand Government website [Working Dogs, 1966], since it is are a country which has a substantial sheep-farming industry, stating that there were 200,000 working dogs there in 1966, handling mainly sheep and cattle. They give a brief description of some of the types of sheepdog that can be used to work sheep:

- Heading Dog – A silent dog that will naturally circle widely around the sheep in order to control the flock.
- Huntaway Dog – A noisy dog that will naturally hunt or chase sheep, more aggressive than the heading dog, used for forcing large groups of sheep towards or away from an area.
- Handy Dog – A combination of the above.
- Leading Dog – Naturally leads the sheep from the front of the flock.
- Backing and Yarding Dog – An aggressive dog, trained to run over the backs of the sheep, used to keep tightly packed sheep moving.
- Stopping Dog – Like a heading dog, but keeps the sheep in a particular location until its master arrives.

This shows that there are several types and personalities of dogs that are used for varying tasks.

A less official source is the Working Sheepdog website, which is run by sheepdog enthusiasts [Nickless and Watson, 2008]. This website gives a good overview of what sheepdogs are expected to do, from the point of view of training a sheepdog. It stresses the point that sheepdogs should move around the flock without disturbing, stressing or harming the sheep. This is particularly relevant since a sheepdog's behaviour is essentially that of a traditional predator, with the gripping and biting part removed [Nickless and Watson, 2008]. From the descriptions of the various types of dogs given above, it is evident that this behaviour is not always entirely removed and can be influenced by the particular personality of a dog. Emotions that may influence this trait include nervousness, frustration, confusion, excitement or fear, since all of these can cause a dog to forget its training [Nickless and Watson, 2008]. The effect of having sheep with different personalities will also be evident here, since a dog may become more frustrated with difficult sheep that may be stubborn to move or be constantly separating from the flock. In these cases the dog may again be more inclined to grip the sheep.

Overall it appears that there tends to be two main categories of shepherding styles, one being more aggressive and the other being more passive. These are identified in sheepdogs as being 'Strong Eyed' or 'Weak Eyed' [Herding Dog, Wikipedia] dogs; *strong eyed* dogs are able to command respect from the flock merely by staring them down, whereas *weak eyed* dogs have to do a lot more in the way of aggression, movement and barking to achieve similar results. Although a *weak eyed* dog sounds less desirable, it is said that they are more suited to working without the supervision of a master.

2.2.3 Predatory Behaviour

As mentioned above, a sheepdog uses a basic predatory behaviour with the kill element removed or suppressed as far as possible. This can be investigated by looking at the hunting behaviours of some other predators that are known to hunt in groups.

Lions

A similar type of behaviour is that of a lion hunting a herd of gazelles, Schaller [1976] looks at the specific predator-prey interactions displayed by these animals. Gazelles do not always flee the moment they see the lion, in fact they stay around and observe the predator until it comes within a certain distance (observed to be 40-60m in the case of lions and gazelles). It goes as far as to say that the prey will ignore 'resting' lions completely. The idea being that so long as at least one animal in the flock can see the predator, at a 'safe' distance, then the flock is safe. When the lion does actually begin to make a move on the prey, the following factors influence the response of the flock:

- Proximity of the lion(s).
- Suddenness of appearance.
- Speed of the lions.
- Number of lions.
- Direction of approach.

Taking these factors into account, the flock will then react by attempting to flee, at this point the group of lions will select a prey that they are going to attack, although Schaller [1976] points out that sometimes the scattering of the flock is enough to confuse the lions, making it hard for them to focus on a single prey and therefore failing to make a kill. It has been observed that lions will adopt a formation whereby some of them will approach the prey from a wider, farther distance in order to encircle it, while others will approach straight on, waiting for the encircling lions to flush out the prey [Barry and Dalrymple-Smith, 2005]. The lions who do the 'encircling' are known as the wings and tend to be lighter and younger, presumably since they need to run further and generally initiate the attacks. The centre lions are older and heavier and are usually involved in the ambush rather than the initial movement. If the prey escapes the ambush the lions can chase for a few hundred metres but eventually they have to stop in order to pant. This is an interesting example of an animal that does not have a speed or endurance advantage over its prey and must instead rely on the coordination of the group's attacks to ensure a kill.

Another example of this coordinated behaviour was that of lions attacking elephants at a watering hole. Joubert [2006] looked at a particular watering hole over several years and observed a consistent pattern. The lions would wait until there were stragglers, usually calves, left behind at the watering hole, calling out for their family. They would then encircle the elephant and then attack from behind (they never attacked front on), from here they would attempt to bring the elephant down for the kill. A second approach was for the lions to simply rush the herd, aiming to cause confusion and bunching, with the possibility of an elephant getting either separated or falling over, leaving itself open for the kill.

From these two main examples it is clear that the cooperation occurs between the lions, without the need for overall planning of the attack. These ideas can be applied to the modelling of multiple sheepdogs, when there is no shepherd to coordinate the sheepdog's movements.

African Hunting Dogs

African hunting dogs (sometimes referred to as African wild dogs), like lions, hunt in packs. Hunting dogs will hunt opportunistically, that is, they will hunt whatever they can whenever they see it. They will often set off in single file before spreading out to cover more ground, allowing them to detect animals such as gazelles lying in the grass, while some hunting dogs will hang back sniffing in thick grass [Schaller, 1976]. It has been observed that the pack can become quite scattered during a hunt, however, the hunting dogs use calls to let each other know where they are, enabling them to re-assemble during the hunt by analysing the direction and distance of the calls.

In the case that they see potential prey before it begins to run away they will stalk (approach slowly) until the prey decides to try and escape [McNutt and Malcolm, 2007]. Once the prey is on the move the hunting dog will chase it until the prey tires or until it gets away. Once the prey tires, the hunting dog can move in for the kill, if other pack members have kept pace during the pace they will join in with the killing. It is not unusual for a hunting dog to make the kill by itself, then return to fetch the rest of the pack to share the kill. In some cases the pack will hunt more cooperatively, for example if the prey is moving in a semicircle to evade one hunting dog, others will move to intercept its path, causing it to become surrounded. Another form of cooperation is that of a *relay hunt* where one hunting dog will take over the chase from another, to prevent the predator from becoming too tired to keep chase [Schaller, 1976].

The cooperative behaviour of African hunting dogs appears to be slightly more complex than that of lions, since it does involve a simple form of communication between dogs during the hunt. The stalking behaviour can be likened to the way in which a sheepdog would approach a flock of sheep without wanting to induce panic.

Wolves

Another animal that hunts in a group is the wolf. Wolves are known to live in strict, well organised social structures, usually centred around a dominant male/female couple [Paquet et al., 2007]. Wolves are believed to communicate through calls, scent and even through the recognition of facial expressions. They hunt in packs, roaming throughout their territory until a suitable prey, often a weak animal or an animal away from its flock is found. Like African hunting dogs, they will chase the prey, but more likely in a group, nipping and biting at it to injure it until they can bring it down for the kill. This relies on the prey getting tired and it not having sufficient speed to escape should it see the predator in time.

2.2.4 Background Research Summary

Since there is much more information available on the hunting behaviour of animals such as lions, African hunting dogs and wolves, it will be interesting to look at how we can model these behaviours and compare them with the current assumptions made about sheepdogs. The most interesting comparison is that of lions, since they have clearly defined formations and methods for cooperative hunting. They actively target separated prey, and although sheepdogs will do the same, it is for the purpose of bringing it back to the herd rather than killing it.

It is also interesting to consider whether or not the cooperative behaviour of the hunting dog, where multiple dogs take different routes in order to surround the prey, can be applied to that of multiple sheepdogs when they are working together to drive a flock. Since sheepdogs have supposedly similar behaviour to their hunting dog and wolf counterparts, we should consider where the divergence of behaviour has occurred and to what extent. From this background research on various animals and their behaviours it is now possible to consider how to model such behaviours and interactions.

2.3 Modelling the Situation

It is clear that there are a variety of questions to answer when it comes to modelling the interaction between sheepdogs and sheep flocks. First of all, how do we create a realistic model of sheep, even without the interaction with predators? Secondly, how is it possible to formalise the behaviours seen in sheepdogs? It may be useful to use some of the analogous animals described earlier to build a more comprehensive model. Finally, we will need to look at a concrete method of implementing such a model on a computer. In the following sections we discuss what work has already been done in building such models and identify any areas that are eligible for improvement or extension.

2.3.1 Modelling Flocks

Flocks, as described earlier, have been modelled in various forms on a computer. The question is how a complex yet elegant behaviour can be modelled realistically on a computer. In this project, it will be possible to use the current work which has largely been related to birds and fish and build on it to create a model for sheep and herding animals in general. This section examines the techniques that have been used so far and it discusses how useful these techniques will be for this project.

Boids

Using computers to model the naturally occurring phenomenon that is flocking is not a new idea. The most widely read and acknowledged paper that first identified flocking as an exciting new idea was the *Boids* paper [Reynolds, 1987]. The fundamental idea behind Reynolds' research is simple, he models each member of the flock as a completely autonomous being that follows three simple rules in order to determine its behaviour. There are no global behaviours or global strategies that govern how the flocks behave; the perceived behaviour of the flock is simply an emergent property of putting more than one flocking entity together in the same environment. This approach was first developed before the idea of Multi-Agent Systems were becoming popular, therefore, a lot of the terminology is different to what we would use today. From now on, we refer to individual elements of a flock as being represented by a single agent, and the system as a whole can be referred to as a Multi-Agent System. Furthermore the concept as a whole can be referred to as Agent Based Modelling (ABM).

The simplicity of the original Boids [Reynolds, 1987] idea can still be used as a foundation for a flocking project, even though it is 20 years old, the idea translates remarkably well into current Multi-Agent System concepts. It will be beneficial to look in more depth at the methods Reynolds [1987] uses to achieve this overall flocking behaviour.

How do Reynolds' Flocks Work?

As mentioned earlier, the basic idea behind Reynolds' Boids is based upon applying three separate 'rules' to each Boid in the system. The system itself can be viewed as a slightly more advanced particle system, in that each particle also has a graphical model and an orientation. Each particle, or Boid, in the system has an awareness of its nearest neighbours in the flock, as in, it is given limited information about its surroundings. It would be unrealistic to provide every Boid with all the information about every other Boid in the flock, as well as being inefficient when it comes to scaling the flock size ($O(n^2)$). Reynolds [1987] talks about how he provides each Boid with the same information that it would receive through its senses, rather than attempting to realistically simulate visual and auditory perception for an animal. The following rules are then applied to the information that the Boid has received about its surroundings.

1. Alignment – Match the speed and direction of nearby flock members.
2. Cohesion – Move towards the average position of nearby flock members.
3. Separation – Move away from nearby flock members to prevent collisions.

Each Boid is concerned with maintaining the above properties within its own 'neighbourhood', there is nothing more to the simulation. Each of the above behaviours are combined into a weighted average that will decide what the Boid should do at any given instant. Reynolds [1987] describes the method for choosing the weighted average in detail, but the basic idea is that in order to prevent Boids from flying into objects, some behaviours are given priority over others.

The elegance of this simulation makes it very easy to observe how small changes to each individual's behaviour effects the emergent behaviour of the group. An approach whereby the entire behaviour of the group was scripted would cause the whole simulation to become inflexible and hard to scale, requiring new code each time a new Boid is added to the simulation.

An interesting feature of Reynolds' Boids is that since each Boid is only interested in its nearest neighbours, rather than the entire flock, the flocks are able to navigate around obstacles without any extra rules or behaviours. This is because as far as each individual Boid is concerned, it is still within its own flock, it has no idea if the flock as a whole is growing or shrinking. Reynolds states that this would not work with a flocking model based on a *central* force or *follow the leader* approach, since in these approaches there is some notion of an objective that the entire flock follows.

Another interesting idea which Reynolds touches on is the speed in which flocks of birds have been observed to change direction. Potts [1984] describes the phenomenon as the 'Chorus Line Hypothesis', in that the speed in which a flock can perform an action such as changing direction, is greater than the sum of the reaction times of each individual bird. This implies that birds and other flocking animals have some perception of when they believe their nearest neighbour is going to change direction and attempt to perform their own change as close to this as possible.

In order to make his simulations more useful, Reynolds [1987] adds the idea of scripting a general objective for the flock and its Boids. For example, he suggests adding a point in the environment or a region where the flock wants to get to, such as flying to a certain region for the winter. This is dealt with in the model by an extra parameter which goes into the weighted average of acceleration change, representing the desire to achieve the objective of getting to a particular place.

Improvements to Boids

Since Boids, there have been various clones and variations on the original idea, the most closely related to this project being another final year project looking at how to implement an agent which can herd sheep to a desired location using flocking rules similar to Reynolds [Pocknell, 1999]. The paper focusses mainly on the emergent behaviour that is seen from using Reynolds' original flocking rules, it has an in-depth description of the various equations and ideas that are required to actually implement such a simulation in 2D; this will be useful when developing the model for this project. The paper does not go as far as to actually implement autonomous control for the sheepdog, but instead gives the user control of the sheepdog to allow us to observe the flock's reactions, depending on how we control the sheepdog. In this project the aim is to create completely autonomous control for the sheepdogs, the methods for doing this are discussed below.

Flocking Summary

The work done by Reynolds in this area is very useful and will serve as a sound basis for modelling the flocking behaviour of sheep. It is clear that there is work to do on adapting the basic three rules of flocking to work for grazing animals such as sheep, who do not always have a desire to flock, and, depending on various factors, who may want to flock more tightly or more often than others. There is potential for looking at the way in which we weight the importance of these rules in order to produce the desired behaviours, perhaps even a dynamic weighting, depending on various emotional factors. At the same time it will be desirable to maintain the simplicity and elegance of Reynolds' [1987] original flocking rules. Once we have developed an acceptable model for sheep flocking, it will be easier to then look at modelling the behaviour of sheepdogs and their interactions with the flock.

2.3.2 Modelling Sheepdogs

As well as looking at a realistic model for sheep flocks it is necessary to realistically model sheepdogs. There has been some previous work on the subject which will provide a good starting point for the model.

One approach to modelling the shepherd behaviour of one or more sheepdogs is to simply treat them as modified flocking animals, where the cohesion and alignment forces towards sheep are increased, making the shepherds simply follow the flock around as closely as possible [Adachi and Kakikura, 2006]. This approach is elegant as it simply draws on the existing framework for representing the flocking animals, however it does not really develop any new or interesting ways of modelling the predator-like behaviour of the shepherd animals and does not look at any cooperative predator behaviours.

A more accomplished approach to the idea of applying flocking rules to predators attempts to model the cooperative hunting behaviour of lions [Barry and Dalrymple-Smith, 2005]. The main ideas focus on using combinations of attractive and repulsive behaviour depending on certain factors, such as the stage of the hunt. Barry and Dalrymple-Smith [2005] claim that it is unlikely that lions are fully aware of other lions during the hunt, since this would suggest that the prey would also be aware of some lions. This is similar to the ideas used by Reynolds [1987], where *Boids* only have knowledge of their immediate neighbours. A set of local rules for cooperative lion hunting is identified as follows:

1. The lions exhibit repulsion/avoidance from the prey to ensure that they are not detected.

2. The lions are attracted to the prey when stalking in order to get within range.

3. The lions are repulsed from other lions in order to cover the widest possible area when closing in on the prey.

4. The lions are attracted to other lions to close off any gaps to prevent prey from escaping.

They claim that using these simple rules, the emergent behaviour would be that of an encircling behaviour. This could be applied to that of sheepdogs, illustrating how sheepdogs might work together in order to keep a flock tightly packed, with the difference being that the dogs do not go through with the kill after the stalking phase.

A more general approach to representing the shepherding behaviours is presented in 'Shepherding Behaviours' [Lien et al., 2004]. They view shepherding as a separate behaviour to flocking and even suggest some real world applications for the study of it such as mine sweeping, surveillance operations or even keeping birds off airport runways. A more concrete definition is provided, describing shepherding behaviour as the attempt by one group (shepherds) to control another group (sheep) by exerting repulsive forces on them. They analysed in great detail the various ways in which a shepherd handles the flock, paying particular attention to the following two aspects of shepherding behaviour.

1. Approaching the flock – How the shepherd will get near to the flock, so that it can begin to steer it.

2. Steering the flock – The techniques that the sheepdog will use to move the sheep once they are in range.

They identify some key concepts that are common across all of the shepherding behaviours which are useful to be aware of when reading the rest of this section.

Steering Points – When a shepherd wishes to move the flock in a certain direction, it will identify a point in relation to the flock, from which it needs to start moving the flock.

Flock Contour – The smallest polygon that can enclose all flock members, used for defining where flock actually *is*.

Milestones – Goals or way-points which the shepherd is aware of, where it must direct the flock to.

Approaching The Flock

Lien et al., [2004] identify three ways of approaching the flock, with varying degrees of complexity and effectiveness:

Straight Line Approaching - This method is as simple as it sounds, the shepherd moves towards the point where it wishes to steer the flock from, in a straight line. It pays no attention to how this might disrupt the flock in the meantime, so it is potentially a bad method, however, it guarantees the fastest route to the steering point.

Safe Zone Approaching – As a direct solution to the above methods' shortcomings, they propose a method where the shepherd does not enter a designated zone, that is, the proximity in which the flock becomes agitated, in this way the shepherd can reach the steering point without disrupting the flock. This is a good idea as it keeps the flock unbroken, but obviously may take longer to get to the steering point, especially in an environment where they may be many obstacles.

Dynamic Roadmap – This is a more complex solution which involves plotting a series of points around the flock, working out how many sheep can see the shepherd if it were at a particular point, and then choosing the route with the lowest counts. This is useful in situations where there are many obstacles or enclosed spaces, meaning that the shepherd sometimes has no choice other than to go near to the flock sometimes.

All three of these methods have advantages and disadvantages, it is clear that an effective shepherd would make use of all of them at some point, depending on the situation. The next point to consider is how the shepherd steers the flock.

Steering The Flock

Once the shepherd has successfully got to the steering point, it then has the task of steering the flock. Lien et al. [2004] describe several methods of doing this, with increasing degrees of complexity and success. A variety of different steering behaviours are outlined in one of Reynolds' [1999] later papers, they focus on how an object moves when a single steering force is applied to it. In the case of sheep and sheepdogs the main steering behaviours seen are seek, flee and pursuit. The descriptions in the paper are very clear and concise and provide a good overview of the mathematics required to implement such behaviours. Lien et al., [2004] go into more detail as to how these steering behaviours relate to shepherd and flock interactions.

Forward Steering Techniques

Forward Steering is described as the technique used when the flock is already heading in the right direction, the shepherd simply needs to add a repulsive force to the flock on the opposite side to the goal. Lien et al. [2004] identified two styles of doing this:

Straight behind the flock - This is the simplest technique which involves simply pushing the flock forwards towards the goal. They observed that this can cause individuals near the edge of the flock to separate from the flock as the shepherd moves towards the centre of the flock. This meant that the shepherd had to constantly move around the flock to gather loose sheep.

Side-to-side behind the flock - A more sophisticated behaviour which Lien et al., have observed to be present in border collies is the side-to-side technique, where the dog alternates between the left and right of the steering point, therefore, keeping the sides of the flock closer to the centre of the flock. Although this seems like it is making the dog cover more ground then necessary, in their experiments it proved to be more efficient, since the flock is much less likely to separate, requiring less chasing of loose sheep.

Turn Steering Techniques

The other type of steering that Lien et al., [2004] identify is known as *turn steering*, this is required when the flock is not heading in the direction of its goal.

Stop-turn steering - The more basic of the techniques that is discussed involves simply stopping the flock when it is heading in the wrong direction. Once stopped, the shepherd can re-evaluate where the steering point should be in order to get to the goal and then start forward steering from that point. This seems to be more of a last-resort technique since plotting a complex path using this would require the flock to be stopped as many times as it needs to be turned.

Pre-turn steering - A better technique is that of pre-turn steering, where the shepherd will make some attempt to recognise that the flock will have to change direction at some point in order to reach the goal. On sensing this, the shepherd can gradually turn the sheep so that their average direction leads towards the goal. Obviously this technique cannot always be used, depending on the environment, as it may be too cramped or contain too many obstacles, so whenever this technique fails it is simple to fall back to stop-turn steering.

Shepherding Behaviours

The most original part of this paper [Lien et al., 2004] is the identification of several different shepherding behaviours rather than just the obvious herding behaviour. The different behaviours that Lien et al., [2004] describe are:

- Herding – The classic shepherding behaviour, taking a flock from point A to point B.
- Covering – Leading a flock round a set of predefined milestones.
- Patrolling – Keeping a flock away from a particular area.
- Collecting – Gathering up separated flock members and bringing them back to the main group.

These different types of behaviours can in fact be linked to the different types of dog described earlier [Working Dogs, 1966], showing that different breeds and personalities of dog may be more suited to different types of shepherding.

Lien et al. [2004] present a great deal of useful information in this paper, providing an excellent foundation for modelling shepherding behaviours. They identify the shortcomings of using a single shepherd to control a flock, hypothesising that multiple shepherds are more efficient, they investigate this hypothesis in a later paper [Lien et al., 2005], which is discussed below.

Perception in Sheep and Sheepdogs

Lien et al. [2004] do not explicitly state how the sheep or shepherds in the model are given their own view of their environment, and we can only assume that it is in a similar way to that of Boids [Reynolds, 1987], as described earlier, where the animals are given a simulation of the information they would get from their perceptions, rather than simulating the perceptions themselves. This may be achieved by looking at the alignment of the animal, taking into account a field of vision and then working out which nearby flock members it is aware of. Conversely, it might be more realistic to simply give the flock members information about all others within a certain radius, which, although similar, allows for the notion of animals remembering their local surroundings.

2.3.3 Modelling Multiple Sheepdogs

As a follow on to their 2004 paper, Lien et al produced another paper examining multiple shepherd behaviour in more detail [Lien et al., 2005]. This built on many of the key ideas from the 2004 paper, but looked in more detail at how multiple agents can accomplish a shared goal without any explicit communication.

The following sections examine the fundamental issues involved in getting multiple shepherds to work together efficiently:

Shepherd Formations

A shepherd formation is how the shepherds assemble themselves with respect to the flock, in order to shepherd them cooperatively.

Line Formations – This involves the shepherds forming a straight line behind the flock, allowing them to distribute the force along the flock, solving the problem of the single shepherd who can only apply force to a certain part of the flock, which risks separation.

Arc Formation – Lining up in a semi-circular arc behind the flock, with the centre of the semi-circle at the conceptual flock centre. The radius of the arc, and therefore the closeness of the shepherds, depends on the *contour* of the flock.

Choosing Steering Points

As described in the previous paper [Lien et al., 2004], the shepherds must choose where to put the steering points. This is slightly more complex in a multi-shepherd system. Lien et al.,[2005] do not actually define how the steering points are decided, it seems as if they rely on some external agent to look at the environment and decide where all of the shepherds should be.

Assigning Steering Points

Once the steering point locations have been decided, regardless of how they were decided, the shepherds must assign themselves to a particular point, without communicating. Lien et al., [2005] identify the following methods:

Vector Projection - Steering points are projected onto the line perpendicular to a line through the centre of the flock. Shepherds are also projected onto this line. Both the steering points and shepherds are sorted so that the first shepherd goes to the first point, the second shepherd goes to the second point and so on. This is a fairly intuitive system, however, if the flock is moving around a lot, the line which the shepherds measure against will move erratically, causing the shepherds to change steering points quickly, and potentially disturbing the flock.

Greedy Distance Minimisation - Shepherds go to their closest available steering points. This again assumes each shepherd knows which points are already assigned. This implies that they are using a third party organiser in their simulations, unless the individual shepherds make best guesses about where their neighbours might go, based on their own preferences. They may even use some sort of predictive method depending on the current movement vector of any nearby shepherds.

Global Distance Minimisation - This uses a more complex formula, graphing all possible combinations and selecting the distribution with the smallest overall distance travelled by all shepherds. Again, this suggests that an external organiser of the shepherds is required, to minimise the distance for the overall good of the task.

Merging Separated Flocks

A useful application of multi-shepherd systems is the distribution of shepherds over separated flocks with the aim of bringing them back together into one single flock [Lien et al., 2005]. This again implies some sort of spontaneous cooperation between the shepherds, based on an implicit understanding of what each other shepherd is doing. Lien et al., [2005] define the three main issues when merging separated flocks. Firstly, they define two flocks as being separated, if all sheep in one flock are unable to see a single sheep from the other flock. However, they claim that this approach makes it difficult for shepherds to keep flocks compact, since the definition allows far apart sheep to be counted as a flock, based on sight range. An alternative approach is that of a *compact area*, which they define as the smallest possible area that a given number of sheep could occupy, the shepherds then strive to keep the flock within this area. The second issue is the need to decide how to distribute the shepherds amongst the flocks, this depends upon the number of shepherds available against the number of separate flocks. Finally, there is the issue of deciding where to actually send the flocks once the dogs are in control of them, for example which flock should become the main flock and what order should the flocks be merged in? This final question is left largely unanswered by Lien at al., [2005] and would be an interesting investigation to carry out for this project.

Multiple Shepherd Summary

Like the previous paper by Lien et al., [2004] this is a very well written paper, clearly presenting their ideas. Although at the core they are still very simple ideas, this can reflect how well suited the area is to computer science, since the original Reynolds [1987] flocking paper was very simple but was extremely well received and respected.

The various questions posed here provide some interesting challenges for implementation; the research is thorough but by no means complete. There is a considerable amount of scope for designing some novel ways of modelling multiple shepherd interactions. Firstly, they do not fully specify the algorithms or techniques they use for deciding steering points, therefore this needs to be investigated. Secondly, they assume that all sheepdogs have the same level of understanding and the same ideas about how to herd sheep, it might be interesting to investigate how different personalities of dog affect overall efficiency. Finally there are many challenges when deciding how to merge separated flocks, when to do so and how to ensure sheepdogs are distributed evenly.

2.3.4 Cooperative Target Organisation

The idea of using multiple shepherds to control or move a flock of sheep is similar to that of the Cooperative Target Organisation (CTO) problem [Luke et al, 2005]. CTO is concerned with ensuring that a number of observers can see the optimum number of targets at any given time. The observers are assumed to have a limited viewing distance, requiring them to work cooperatively in order to observe the maximum number of targets by distributing themselves efficiently. There are two general approaches, the centralised approach whereby there is a single process that gathers all the information about the process and then distributes the observers optimally. If we were to look at how a shepherd controls his sheepdogs, this would be an interesting analogy. However, if we are just looking at how sheepdogs work together without any central controller and without any direct communication, we would look at the other approach, the de-centralised method where agents decide what to do for themselves based entirely on local information.

The paper describes two algorithms for finding the optimum distribution of observers. Firstly hill-climbing tries random assignments of observers across the environment, then incrementally moves each one to try and improve the overall observation coverage, this continues until no significant improvements can be made. The second algorithm, k-means, is a little more complex, it is a generic clustering algorithm which attempts to assign centres of clusters such that the overall distances between targets and their observers are at a minimum [Hartigan, 1975]. The algorithm works as follows, with K clusters (centres) and N points:

1. Assign each cluster K some initial position, potentially random.
2. Attach each point N to its closest cluster K.
3. For each cluster, calculate the mean centre, based on the N points that are assigned to it, and move the cluster to the mean position.
4. Go to 2, with new cluster positions

This algorithm continues until a certain level of convergence is reached, whereby the clusters cannot be moved to improve the efficiency of their distribution. The rest of the paper describes how altering the degree of centralisation affects the results, this is not particularly relevant to this project, since we will be implementing decentralised sheepdogs. The ideas presented for the de-centralised approach provide an interesting motivation for the design of multiple sheepdog interactions.

2.3.5 Shepherding Summary

From the above discussion, it is clear that there are two main ideas for modelling shepherds or more generally, predators. We can extend the flocking rules of Reynolds [1987], as suggested by Barry and Dalrymple-Smith [2005], to make the predators use combinations of attraction and repulsion to other predators and the prey. This has obvious benefits in that it extends from the model that we will be using for the sheep flocks and would provide an elegant way for the sheepdogs to interact with the flock. The other approach, as outlined by Lien et al., [2004, 2005] looks at the behaviour from a different angle, largely ignoring the work of Reynolds [1987] and instead analysing the behaviour of shepherds in a more general sense. It will be interesting to look at both of these methods to see which produces the most realistic and believable results.

If we find that the Lien et al., [2004, 2005] way of implementing shepherding behaviours is the most effective, it is clear that there will be a reasonable amount of work needed to devise effective algorithms for *approaching* the flock and assigning *steering points*, as well as deciding how to distribute sheep and sheep flocks across multiple sheepdogs. The four main shepherding behaviours identified by Lien et al., [2004](Herding, Covering, Patrolling, Collecting) will provide a wide variety of situations to consider when testing the effectiveness of the sheepdog behaviours implemented in this project.

The algorithms used for Cooperative Target Organisation could be used as a way of assessing where the centre of flocks are. This would be useful for sheepdogs, allowing them to work out where to send loose sheep and giving them a point of reference for steering a flock as a whole. This could go some way to solving the problem of deciding where steering points should be placed.

Overall, there is a considerable amount of scope for experimenting with new ways of modelling multiple sheepdogs interacting with sheep flocks. The following section identifies the tools that are available for achieving such models.

2.4 Agent Based Modelling

Although not specifically referred to in Reynolds' [1987] original paper, it is clear that the ideas that he used were similar to what is now known as Agent Based Modelling, or Multi-Agent Simulations.

Agent-Based Modelling (ABM) is a relatively new area of Computer Science, designed for modelling complex systems using a bottom-up approach. It is often used to model social processes but is also known to be used to model ecology, biology, anthropology, psychology and traffic and vehicle simulations [Gulyás, 2005]. ABM usually consists of an environment, such as a model of a spatial environment or indeed something more abstract such as an agent trading community [De Vos and Padget, 2007]. An environment may have its own behaviour, making it dynamic and allowing it to co-evolve with the agents that inhabit it. Within the environment are the agents themselves, they exist completely independently of one another, acting autonomously based on their own set of rules and perception of the environment. Already from this brief description it is clear how similar Boids is to an ABM today, even though it is far older than the field of ABM recognised today.

The overall properties of the system can be described as the emergent behaviour, that is, the overall macro level behaviour that arises from multiple agents interacting with each other and the environment. We can use ABM to experiment with a variety of parameters in a simulation with very little effort. For example we could measure how many steps it takes for a dog to herd sheep together, performing the same experiment for 10, 20, etc. sheep without writing any new code. This type of investigation is known as a *parameter sweep* [De Vos and Padget, 2007].

2.4.1 Precursors To Agent Based Modelling

One clearly identified precursor to ABM is *cellular automata*, which is a more simple idea, based on a finite state automaton. The idea is that we divide up an environment into a grid, or a group of cells. Each cell has a state, usually represented by a number. Then, on finite ticks of a clock, each cell will alter its state according to its finite automata, based on its nearby cells. It is clear how ABM evolved out of this sort of idea, cellular automata does have the idea of individual cells acting autonomously, but does not capture the same sort of freedom that an ABM implies, since the cells do not 'move'.

The archetypal example of a cellular automata is the Game Of Life [Gardner, 1970]. It is a zero-player game that simulates the birth and death of organisms in a simple grid, or, a cellular automata. Like the flocking algorithms previously described, it uses a few very simple rules to create interesting emergent behaviours, allowing the 'player' to experiment with many different initial configurations, observing how they play out when the simulation is run.

A relatively new paper [Adachi and Kakikura, 2006] has looked at modelling interactions between sheepdogs and sheep using cellular automata. It basically looks at implementing Boids using cellular automata, without giving a reason why they chose that particular method. The paper does show that this method is more suited to more formal mathematical representations of individual actions in the system, whereas ABMs are more flexible and can define behaviours and actions in whichever way the programming language of choice allows.

2.4.2 Equation Based Modelling versus Agent Based Modelling

Another alternative to ABM is Equation Based Modelling (EBM). Like cellular automata, EBM is a more formal approach to modelling real world situations, by devising a set of formal equations that describe the relationships between individuals and then evaluating or integrating these equations [Parunak et al, 1998]. This is different to ABM which instead focusses on how the individual behaves, leaving the interactions between individuals to be represented as emergent behaviour. EBMs tend to model individuals as homogeneous entities, whereas ABM gives us the potential to give each individual subtly different behaviours.

A drawback of EBM is the complexity that is associated with larger models. Since the equations describe the system-level observables, as the system grows the complexity will also increase. Compare this to ABMs where adding new participants to a model is simply a case of instantiating another object. On the other hand, an advantage of EBM is that due to its formal nature, it is more suited to modelling large numbers of individuals, since it can be encoded as a mathematical formula. In an ABM an increase in the number of individuals requires a noticeable amount of processing power, to model each individual's actions concurrently.

ABM is not without its disadvantages. It can be said that obtaining the right level of detail in a simulation is an art rather than a science [Bonabeau, 2002]. Being able to find enough information in a given domain in order to model it correctly can be difficult, but ensuring that the right amount of information for the intended purpose is used can be just as hard. Another problem can be that since we are often modelling social situations, be it around humans or animals, behaviours are often affected by irrational decisions or complex psychological processes. Because of this it can be tempting to engineer agents to exhibit some desired overall behaviour without actually getting the behaviour at the micro level correct. Therefore, it is important to understand the domain we are modelling to an extent that allows us to correctly model the low level behavioural rules.

For this project an ABM is the preferable model since it allows us to model a heterogeneous population of agents, built from the ground up. An equation based approach would require most of the project to be spent formalising a complex model with very little room for changes along the way during development.

2.4.3 Agent Architectures

The discussion on what architectures exist to model agents is ongoing and there is by no means a concrete 'best' way to create an agent to model a particular real world entity. The four main types of agent architecture that are recognised today are: deductive reasoning agents, practical reasoning agents, reactive agents and hybrid agents [Wooldridge, 2002]. It is not necessary to discuss each one in detail, however, it is useful to briefly discuss some of them.

A *practical reasoning a*gent reasons about its beliefs, desires and intentions and formulates plans to try and achieve its intentions, taking time to deliberate about whether or not particular intentions are still possible, whether or not they have been achieved and if the agent still holds that particular intention. For this project, it is clear that a sheepdog could be modelled using a similar architecture, with the intention being to round up sheep. However, it will be interesting to see whether or not a realistic sheepdog behaviour could be implemented by using a purely reactive agent, without the need for complex reasoning algorithms. This would mean that the sheepdog agent would not maintain any state and would base its actions entirely on its perceptions at a certain point in time. A potentially better idea would be to use a hybrid architecture which allows the use of various architectures, organised into layers. For example, a reactive layer can be included to deal with behaviours such as collision avoidance, while a practical reasoning layer could be used in parallel to implement the more long term behaviours and goals.

As we have seen already, Reynolds [1987] created a simulation which relied entirely on reactive behaviour. It achieved this by mapping inputs to a series of outputs and taking a weighted average of these outputs in order to produce a single output. An alternative to this would be to use Brooks' Subsumption Architecture [Brooks, 1986], which instead uses a layered approach of behaviours. For example avoiding collisions would be the lowest level behaviour, and the next layer up might be avoiding predators, all the way up to the top layer which may be moving randomly. Lower layers inhibit higher layers, such that an agent will always perform the lowest action that it can match with a series of preconditions. As seen with Luc Steel's Mars Rover [Wooldridge, 2002], seemingly complex behaviour can emerge from a simple set of layers. It will be interesting to see if this approach is appropriate for this project, or if the layering model will become too complex as we try to implement more complex behaviours.

2.4.4 Comparisons of ABM Frameworks

There are a variety of frameworks written in various languages that provide a starting point for developing Agent Based Models. The available frameworks range from those such as Netlogo that make it possible for beginner programmers to create Agent Based Simulations, through to more complex and comprehensive frameworks such as MASON and Repast [Railsback et al., 2006].

It is not necessary to get into a detailed comparison of every major ABM framework here, however we should consider some of the important factors when choosing an ABM. It is important to consider the programming experience required to use an ABM. NetLogo is the simplest and is regarded as the easiest to use, albeit at the expense of being somewhat limited, for example NetLogo can only work with grid environments whereas most other ABM frameworks can use continuous environments. Swarm was originally written in Objective-C and is one of the older ABMs. There was demand for it to be re-written in Java, presumably because many people felt this was an easier language to develop with. Repast again builds on Swarm and uses Java, while MASON was built from the ground up to use Java. Overall it seems that the most functional ABMs use Java, which is acceptable for this project.

Of the available Java platforms, MASON turned out to be the fastest [Railsback et al., 2006], and although this is not a major consideration for us, it is useful to consider. Objecive-C swarm was unsurprisingly the fastest overall. Another factor is the ease of development, the Java based ABMs will be able to integrate with an IDE such as eclipse, aiding coding and providing sophisticated debugging tools. NetLogo uses its own IDE for this purpose.

The overall view from Railsback et al, is that MASON is a good choice for a more experienced programmer, whereas NetLogo is the most obvious choice for an inexperienced programmer. Therefore, it makes sense for us to develop our project using MASON.

2.4.5 Summary of ABM

As mentioned in the background section [section 2.2], different breeds of sheep exhibit different personality traits, and even sheep within the same breed can react differently in similar situations. These different personalities provide an interesting case for modelling the sheep flock as a group of heterogeneous agents rather than homogeneous agents. ABM makes it easy to integrate many different behaviours of the same type of animal into one environment.

Having looked at the benefits of using ABM for this project, we are presented with an interesting question as to what type of agent architecture to use. It will be interesting to analyse whether or not a purely reactive agent is enough to model sheepdogs, or if a more complex practical reasoning or hybrid agent will be required. Similarly, if we can implement sheep using purely reactive behaviour, will it be possible or beneficial to use an existing architecture, such as the Brooks Subsumption architecture [Brooks, 1986].

2.5 Emotion and Personality in Agent Based Systems

Carrying on from the ABM discussion, a way of potentially modelling the above shepherding behaviours might be made possible or enhanced by looking at emotion, personality and mood. Since there has been little research in this area it is possible to examine the work done so far in chronological order.

2.5.1 Spontaneous Coordinated Behaviour

One of the earlier papers [Shibata et al., 1996] looks at “Spontaneous Behaviour of Robots for Co-operation, using Emotion”. The paper looks at cooperation between physical robots which can be interpreted as agents for the purpose of this project. *Spontaneous coordinated behaviour* is described as more than one robot working together without any supervisor instructing how to work together or optimising the task. This is analogous to the idea of multiple shepherds working together to accomplish a task without having any communication or any leader telling them what to do (assuming there is no master).

The general idea of the paper is about how the robots choose a strategy to tackle a particular problem and then says that they may change strategy depending on the emotions they feel. They use an evaluation function to determine how successful a strategy is at a given point (e.g. how much time has passed, how complete the task is) and then if the strategy is not working well, they increase the frustration emotion. The fundamental idea being that the robot will change its approach based on its frustration level, potentially breaking deadlocks and improving efficiency. Overall, this paper merely touches on the idea of using emotions to control the behaviour of autonomous beings, but without a comprehensive method of doing so. One interesting thing it does bring up is the idea of spontaneous coordination, seeing the robots working together without any communication, since this type of behaviour is extremely common in the animal kingdom.

2.5.2 Emotion to avoid Deadlock

Another way of using emotion is as a way of avoiding deadlock between several cooperating, heterogeneous agents. Murphy et al., [2002] performed experiments that involved robots performing an interdependent task. The robots were based on a hybrid architecture, including an emotional layer used for modifying the set of active behaviours. They approached the problem by using a multilevel process theory of emotions [Leventhal and Scherer, 1987], which consists of three levels:

- Sensory-motor level – Emotions modify the output of active behaviours, perhaps the speed or intensity of an action.
- Schematic level – Emotions modify which behaviours are actually in use.
- Conceptual level – Reasons about previous experiences and emotions and deliberates about what to do in the future.

The aim is that between heterogeneous agents, there will be an emergent societal behaviour, preventing issues such as deadlocks. They focus mainly on the sensory-motor level and the schematic level, proposing that the schematic level can be implemented by a series of scripts, basing its outputs on a measure of how complete the task is. They do not look at how agents' emotions may be influenced by a given third party when it is not directly related to the progress of a task.

Murphy et al., [2002] use the idea of having some overall state of the agent, for example happy or confident. This will have some bearing on the behaviours chosen and the way in which the behaviours are carried out such as the parameters that go into the actions selected.

Overall they are looking for new ways to control agents, looking to the natural model of emotion as an abstraction for developing action selection mechanisms. They argue that the emotional model is a step forward from the traditional control methods. The main advantage is that of breaking cyclic dependencies, or interlocks, without the need for some centralised agent taking care of ensuring no deadlocks occur. They describe the emotional style abstraction as being positive in that people who may interact with the agent or robot have some preconceived expectations of how people may behave based on certain emotions, meaning that they can respond more appropriately, even to robots. Finally and most notably, they point out that they were able to implement the desired behaviours using emotion in far less lines of code than that of traditional, modularised, behaviour mechanisms.

We can draw parallels here with the ABM versus EBM argument. ABM produces the desired overall behaviours through emergent behaviours, using less complex code than that of EBM, and similarly emotional control can bring about the same results as traditional code, using emergent societal behaviour. In the same way however, it can be argued that this makes the system of emotionally controlled robots even harder to test, since we are working with an emotional subsystem, relying on the emergent behaviours rather than having strict behavioural modules to analyse. It is observed how it is not necessary for agents to understand what emotions other agents currently have, it is enough for them to observe how the other agent is behaving based on those emotions, and to act accordingly.

Murphy et al. [2002] did not look into the conceptual level of emotions, since they claim this to be too complex to model. Also, since they only experimented using two robots, they are unsure whether the emotional model they proposed will scale particularly well, especially since it relies heavily on emergent behaviours. However, if we compare this to the idea of ABM, the prospect seems promising since the underlying idea behind ABM only relies on understanding the micro level interactions, with the overall behaviour emerging from these interactions.

This is a good paper that provides some possible directions to explore, for designing a model for sheep and sheepdogs using emotion. The simple layered model makes it easy to imagine how the personality and mood of an animal may affect the actions it selects, while its short term emotions will affect how those actions are performed.

2.5.3 Emotion on Action Selection

The idea of emotion being a mediator between individual behaviour and group behaviour was looked at by Delgado-Mata and Aylett [2004]. They identify emotion as being a form of short term memory that affects Action Selection. They describe sheep as having competing behaviours, the desire to be within a flock and the desire to graze. The emotion they describe is a means of preventing the sheep from dithering between the two behaviours. The emotion they use in the paper is fear, in their model, sheep communicate fear by pheromones, so that they can sense when a predator is nearby. When they sense fear their desire to flock increases, thereby decreasing the desire to graze.

This simple use of emotion is a good example of how much could be done with a more in-depth emotional system, and later on the idea was revisited by Delgado-Mata and Ibanez-Martinez [2006]. They decompose the 'brain' of an animal into a sequence of *Perception, Action Selection and Motor Control.* Their idea of emotion adds another part to the sequence, *Emotion* receives *Perception* information and affects *Action Selection*. The paper then discusses some of the ethological ideas and models that their system is built upon, this discussion is beyond the scope of this project, we will instead look at how Delgado-Mata and

Ibanez-Martinez [2006] applied their research to this subject. They again focus on the fact that grazing mammals such as sheep will spend 80% of their time grazing and hypothesise that emotion is a good way to model this behaviour. They use a table of state transitions with probabilities assigned to them to represent the animals behaviour, the interesting part of this is that they use emotional state as an input to the transition function. In this way they claim to be using emotion as a form of short term memory, they extend the traditional flocking rules with emotion in order to give each sheep a more natural and individual behaviour.

The actual experiment that they focus on involves comparing 'Rigid Flocking' in which all animals are tightly packed in a herd and move uniformly, with 'Flocking', in which the classic Boids [Reynolds, 1987] rules are applied, and a control experiment of random movement. They also experiment with giving the animals an 'Escape' action in which they avoid predators. The interesting comparison which they make is between the Flocking with Escape behaviour and the *Emotionally Controlled* behaviour. They observe that both provide realistic flocking and avoidance of predators, however the *Emotion-based* model appears to show more organised movements. They assume that the animals are communicating fear through pheromones, which causes their movement to be more organised, since the sheep are able to perceive threats more consistently using the emotional information.

This paper will be useful to consider when implementing this project, it will be interesting to see whether or not emotion can indeed make the sheepdog and sheep interactions more believable. It would also be beneficial to investigate whether any advantages can be made by using emotion as described in this paper without modelling communication through pheromones.

2.5.4 Emotion, Mood, Personality

An area of flocking simulations that has not been covered quite so well is the consideration of what makes each member of a flock individual, that is, their personality traits and emotional state. The idea of representing emotion and personality in an agent is not an entirely new idea, it has been looked at for use within an Interactive Learning Agent [Li et al., 2007] in order to try and make these learning agents appear more human like.

This paper touches on some very good ideas and presented some thoughts that were applicable to many different areas of artificial intelligence. They present an established theory for modelling personality called the OCEAN theory of personalities [Wiggins, 1996] as well as the OCC model [Ortony et al., 1998] which uses this theory to model the emotions, mood and personality of a being. As mentioned earlier, it is outside the scope of this project to delve into the details of psychological models, instead we will mention them and focus more on how they are applied to the Computer Science based application in question.

This paper does not focus on action selection, but instead on how the user of the system perceives the agent. They choose a subset of emotions and apply weighted averages to them and combine them with a 'previous' value of mood to determine the agent's' current mood. The agent's personality affects what different emotions may cause the agent to do. They view mood as a more long term form of emotion, whereas 'emotions' are relatively short term.

It will be worthwhile to use both the long term ideas of mood and short term emotions described in this paper to see if it helps make the interaction between sheep and sheepdogs, and also between sheepdogs themselves, any more realistic. However this paper offers little more since the application of emotion is mainly aimed at making communication with an agent more realistic and human-like.

2.5.5 Emotion Summary

It is clear that there is no sole recognised method for representing emotion, mood and personality in an agent based model. Of the papers discussed above, there is very little crossover in method or technique. Although there are other papers available, there were none that could add anything new or interesting with respect to the above papers.

The idea of applying emotion to animal interactions seems natural, since we are using our own knowledge and understanding of our own emotions and using them as an abstraction for general animal emotion. Whether or not this will be an effective model remains to be seen, we will need to try to implement an effective emotional subsystem that does not break down the flocking and shepherding behaviours implemented without emotion.

From the background research to sheep and sheepdogs, it is clear that the subject area has potential for modelling emotion. Sheep of different breeds are seen to have different personalities, and individual sheep have been observed to get stressed more easily, panic, rebel more frequently and have varying desires to be with the flock. An emotional subsystem in an ABM will provide an interesting platform for modelling these differences between various types of sheep. It may also be useful to look at different personalities of sheepdog, such as the *Strong Eyed* and *Weak Eyed* dogs as described in the background reading. Since we believe that sheepdogs do actually exhibiting some form of modified predatory behaviour, it is possible that the extent to which the predatory behaviour has been modified can be affected by personality. Moreover, since some dogs are known to grip difficult sheep, we could model this as the dog becoming frustrated or over-excited, which is clearly an emotional response.

Finally, there is the idea of using emotion to make the simulation more realistic by modelling more long term interactions, such as the effects of a sheep getting scared or angry over a period of time and observing how a sheepdog copes with this. Without the emotional model the interactions can be seen as one-shot encounters, each one depending only on the current state of the environment and the position of the animals. This links back to the idea of emotion being a form of short term memory [Delgado-Mata and Aylett, 2004]. We can also look at the effectiveness of dogs with different emotional balances or imbalances, for example, examining if a “good” sheepdog is one with a particular set of emotional traits.

2.6 Conclusion

From this research it is clear that the original work by Reynolds [1987] was very well respected and provides a good starting point for this project. As previously mentioned there are several potential improvements that we can make to the original work, such as adding new rules or tweaking the weightings to tailor the model to sheep and sheepdogs. The work on using an extended set of flocking rules to model hunting provides a wider range of possibilities to experiment with, based on Reynolds' original work.

The various papers on shepherding [Lien et al., [2004, 2005]] and its different approaches have given us a lot of potential for investigating how shepherd animals can work with flocks in different ways, and potentially for looking at how a shepherd animal might adapt to different strategies depending on the flock. There is great deal of work to be done on looking into how multiple shepherds can work together to achieve a common goal, without any explicit form of communication (this is similar to the general flocking algorithm, where simple internal rules result in a more complex group behaviour). The research on predatory behaviour other than that of the sheepdog will allow us to draw comparisons between the behaviours and to investigate where this modified predatory behaviour that sheepdogs appear to exhibit has come from.

The challenge of actually implementing these documented behaviours is a significant one, there are many technical considerations such as choosing appropriate algorithms to model the perception and thought processes of the animals. It is clear that ABM is a very suitable methodology for investigating the interactions between sheep and sheepdogs. We look at various agent architectures available for implementing the system, seeing if there are any with definitive advantages for this particular domain.

The research into how personality and emotion can be represented in agent based simulations give us an interesting direction to take, especially when modelling the flocking side of the simulation. An emotional representation will be useful when trying to model the different types and characteristics of sheepdogs described in the background research. The idea of modelling sheep as homogeneous agents, but using personality, emotion and mood to make them appear heterogeneous is interesting. There is much more to investigate in terms of the effects of emotion on action selection, using emotion to break deadlocks and using emotion to make the simulation appear more realistic. Since we have identified the possible link between natural predatory behaviours and that of sheepdogs, it will be useful to see whether or not it is the emotional systems that are causing this divergence from natural behaviour.

In conclusion, it seems that while flocking is a well studied subject, there is still scope to improve on and discover more about shepherding behaviours, especially multi-shepherd behaviours. It is also clear that there is no real mainstream acceptance for any form of modelling emotion within an agent and therefore we will have a lot of scope to make an interesting model based on emotions and personality.

Chapter 3, Research Objectives

The most general objective was to investigate the interaction between flocks of sheep and sheepdogs. This will involve attempting to model the sheep flocks and the behaviours of the sheepdogs, looking at how they may cooperate and the various techniques that may be used in order to herd and gather sheep. We can define some more specific objectives:

1. Investigate whether Reynolds' [1987] flocking rules are sufficient to model the behaviour of sheep, or if extra methods and techniques are required.

2. Investigate how to model the reactions of sheep towards one or more sheepdogs.

3. Investigate how to model the behaviour of a sheepdog, this will include techniques to gather and to herd sheep, as well as covering flocks and preventing them from disbanding [Lien et al., 2004].

4. Investigate the cooperative behaviour of multiple sheepdogs, while achieving the above tasks. This will involve spontaneous coordination [Shibata et al., 1996], similar to that of predators such as lions when hunting [Barry and Dalrymple-Smith, 2005].

5. Investigate ways to model different sheep personalities such as leader sheep, light sheep and heavy sheep [Downe, 2005]. These different personalities can then be used to see how the flock dynamics may change, as well as how the sheepdogs tasks become either easier or more difficult.

6. Investigate the use of emotion to vary the behaviour of sheep, depending on their situation, personality and past experiences.

7. Investigate the use of personality and emotion in sheepdogs to model the different techniques that dogs use such as using 'eye' [Downe, 2005].

Due to the broad nature of these objectives, it was unclear whether or not objectives 6 & 7 would be investigated thoroughly. We follow these objectives throughout the various design iterations and perform tests to assess the extent to which we have met the objectives. In the conclusion (Chapter 9) we look back to these objectives to see what we have achieved throughout the project.

Chapter 4, Design

This is an Agent Based Modelling project, therefore the traditional "*requirements-design-implementation-testing*" layout was not particularly appropriate, since a lot of the work focuses on iterative design of both the sheep and sheepdog models. As the models were implemented, emergent behaviour [Wooldridge, 2002] was seen, which was found to be difficult to design for, predict and debug. As a consequence, this section first looks at some of the broader design ideas relating to the simulation. Following that, we focus on the design of the sheep model, discussing what it is required to do, the considerations made during the design, as well as difficulties found or improvements made during implementation. With the sheep model dealt with, the design process of the sheepdog model is described, discussing the two main types of dog that were implemented as well as the findings and new ideas discovered along the way. Finally we discuss implementation details or decisions that did not fall in to the natural flow of the sheep or sheepdog design process, this goes as detailed as class diagrams and some pseudo code snippets; full, commented source code can be found on the attached CD.

4.1 Environment

The model used a 2D environment, each animal was represented as a point in this environment. Animals have an XY coordinate associated with them, as well as a level of momentum. The environment was bounded within a particular width and height, and was not toroidal, that is, it did not wrap around at the edges. Movements and forces were represented as 2D vectors.

It was necessary for sheep to avoid the edges or 'walls' of the environment, to prevent flocks from getting 'stuck' to walls. An imaginary rectangle, slightly smaller than the environment was used to enclose the sheep, but not the sheepdog. This meant that the sheepdog could always 'get behind' the flock, allowing it to concentrate on herding the sheep, without having to consider how to move flocks away from walls.

4.2 Animal Locomotion

A fundamental area of this project was the locomotion of the animals being modelled. This involved modelling a realistic way in which the animals will move, within the 2D environment.

On each step of the simulation, a steering force is calculated based on the animal's flocking rules or on another action selection mechanism. The steering force is weighted against the current momentum to produce a new overall movement vector. The aim of this was to make turning more realistic, since animals are not able to change direction instantly.

The weighted combination of the animals' momentum and the steering force was parameterized by an *agility* value for the particular animal. An animal with high agility is able to turn more easily, while an animal with low agility has to perform a wider turning circle.

4.2.1 Desired Location

Each animal makes a decision as to where it would rather be in the environment, based on its current state and surroundings. This is represented as a vector from the animals current location to the desired location.

Normalisation

Each animal has an associated maximum speed, that is, the maximum distance they can move per step of the simulation. Therefore, when calculating where to move to, the vector to the desired location must be normalised, such that the magnitude is less than or equal to the animals maximum speed.

Momentum

Once the animal has a normalised objective vector, it is then weighted against the movement from the previous step of the simulation, or its *momentum*.

```
movement( objective vector, previous movement ) =
                              a * objective vector + b * previous movement.
```

The value of a and b should add up to 1.0, and depend on the animal in question. For example a dog is generally more agile than a sheep, so a dog may have the values of a=0.8 b=0.2, while a sheep may have the values of a=0.5 and b=0.5.

4.3 Animal Perception

The decision was taken early on to use a simple animal perception mechanism, to allow the project to focus on the behaviour and interactions of the animals. Sheep are prey animals, with their eyes on the sides of their heads, allowing them to see everything around them with minimal movement [Gill, 2004]. Using this 360 degree vision idea, the sheep were given a local neighbourhood, specifying a circular area around the sheep in which the sheep can sense other animals. The vision for dogs was also simulated this way, giving them 360 degree vision, with a far wider field of view.

4.3.1 Animal Neighbourhood

The neighbourhood of an animal is the circle, centred at the position of the animal, which encloses all other animals close enough for the animal to 'see'. As described above, this project simplifies the vision of an animal into a full circle, with a particular radius. The larger the circle, the earlier an animal can detect others, for example, a sheep with a larger neighbourhood can detect and flee from predators sooner.

4.4 One Man and His Dog?

In a traditional sheep herding situation, there is a shepherd controlling his dog, or in some cases, multiple dogs. This project does not consider the role of the shepherd in the herding, it was decided that the dogs would act independently, without any external coordination. Since the dogs work without a shepherd, we assume that they are relying on their natural instincts. It has been observed that during herding when the dog is too far away to see or hear the shepherd's commands, it attempts to herd the flock back into the shepherd's line-of-sight [Downe, 2005]. Therefore, the design of this simulation assumes the sheepdog is acting on a natural instinct to drive the flock to a particular location.

It became apparent that some centralised coordination of tasks would be needed, in order to make the simulation work at all. For example, to perform any sort of gathering or herding task, the sheepdog must have an idea of where to send the sheep. When multiple sheepdogs work together to herd a flock to a particular destination, this destination must be specified elsewhere and given to each dog. This led to the idea of an *oracle* [section 6.6], or, some entity which informs the dogs of their goals. In the same way, it was necessary for *the oracle* to check when goals had been achieved and inform all of the sheepdogs.

To make it possible for multiple sheepdogs to work together, they were required to have their own local behaviours which allow cooperation to arise as an emergent behaviour. The hunting methods of lions, as discussed in the literature review, demonstrates cooperative behaviour without a designated coordinator; the lions cooperate based on their local view of the environment [Barry and Dalrymple-Smith, 2005]. Several techniques for achieving this type of coordination are described in the sheepdog design section [Chapter 6].

Chapter 5, Sheep Model Design

The following section outlines the key points from the background research that were considered when developing the sheep model.

When in the presence of at least four other sheep [Weaver, 2005], sheep exhibit flocking behaviour. Additionally, sheep react to a nearby predator by forming flocks, no matter how small their number, and flee from a predator if it enters the sheep's flight zone. When alone, with no natural predators, sheep simply wander around the field [Smith et al., 1997].

After a sheep has been pursued, it remains in a nervous or scared state proportional to the amount of time for which it was pursued [Downe, 2005]. This means having a desire to flock with nearby sheep, even after the threat has dissipated, until its level of fear has died down. A sheep balances its time between flocking and wandering with the desire to graze, in order to ruminate [Delgado-Mata and Ibanez-Martinez, 2006]. Intuitively, a sheep only feels comfortable enough to graze when under no immediate threat from a predator or when in the company of enough other sheep [McFarland, 2006 ("Herding")].

Some breeds of sheep will stand up to predators more often, therefore the model includes light sheep which are very timid and flighty [Downe, 2005], as well as heavy sheep which are more stubborn. These effects can be characterised by a sheep's feeling of fear. The amount of fear a sheep experiences will be modelled based on how many predators it is near to, and the aggression of these predators.

Sheep can be influenced by other sheep in varying ways, this is represented by a leadership rating for each sheep. Flock members are more likely to follow or stay near to a sheep with a higher leadership rating. Sheep with higher leadership values are more likely to leave the flock and less likely to flee from predators. On the other hand, weak sheep are more likely to be ignored by other flock members and make less effort to get to the centre of the flock [Sheep, Animal Corner, 2008].

5.1 Sheep Model Iteration 1: Sheep Flocking Behaviour

5.1.1 Purely Reactive Agent

The initial sheep model was implemented as a purely reactive agent, making decisions on each time step of the simulation based purely on its current perception of the environment.

The key point here is that the sheep does not maintaining any state or memory of what has previously happened. The sheep makes decisions on how to act by taking a view of the environment and then applying the various flocking rules to decide how to act.

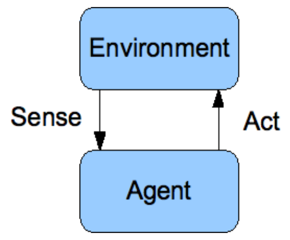


Figure 1: A purely reactive agent

5.1.2 Implementing Flocking Rules

The sheep model used Reynolds' [1987] original flocking rules as a starting point. Reynolds used the idea of each individual having a neighbourhood; when calculating each steering behaviour, only other individuals within the neighbourhood were considered. To implement the three fundamental rules: separation, cohesion and alignment, we considered the following ways of scaling the resultant vector for each neighbour.

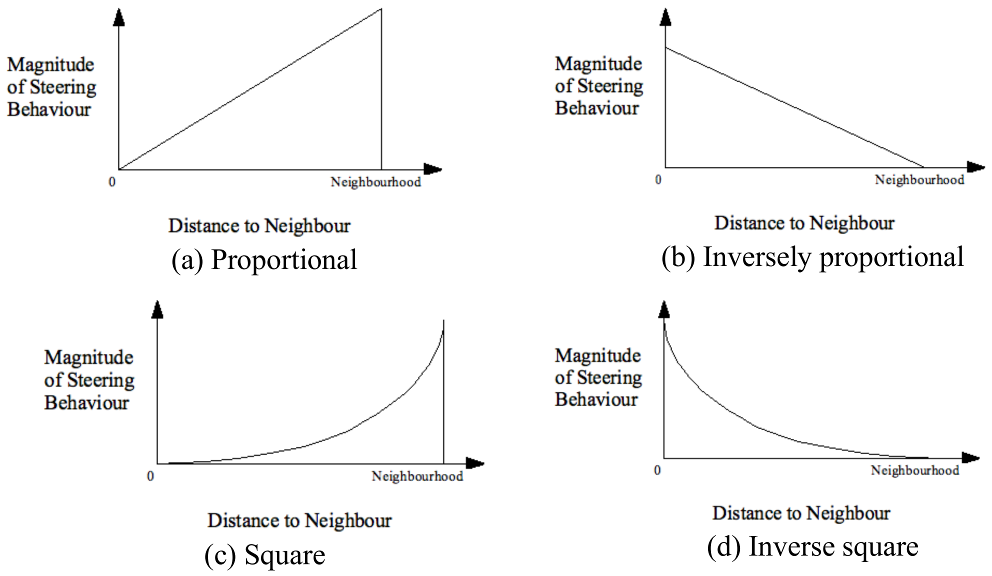


Figure 2: Scaling methods for steering behaviours

Using these ideas (fig. 2), each of the steering behaviours is described in more detail below, including additional behaviours for fleeing from predators and adding an unpredictable element to the sheep behaviour.

5.1.3 Alignment

This rule represents the individual's desire to move in the same direction, at the same speed as the rest of the flock. In the case of birds, this is an important rule, since birds must be moving when they are flying, therefore it is important for them to coordinate their motion. In the case of sheep, it is not so important for sheep to always be sharing the same direction and speed. The algorithm for alignment is as follows:

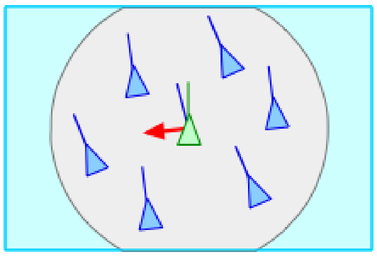


Figure 3: Alignment [Reynolds, 1987]

- For each Sheep in the local neighbourhood:
    1. Get the momentum of each nearby sheep.
    2. **Inversely scale** the momentum based on the distance from the sheep.
- Return the average of the sum the scaled momentums.

5.1.4 Cohesion

The cohesion steering force represents the desire of an individual to be in the centre of the flock. Intuitively, this is key rule to sheep flocking. The desire of each sheep to be at the centre creates a tight flock.

This is achieved by looking for a sheep within a certain *sight range*, and then moving towards the average position of these sheep. The further the sheep is, the greater the magnitude of the attraction. The algorithm for cohesion is as follows:

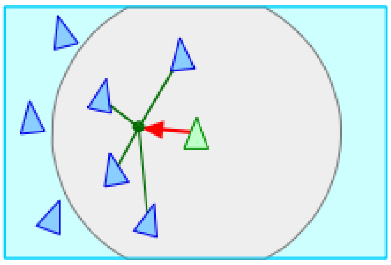


Figure 4: Cohesion [Reynolds, 1987]

- For each sheep in the local neighbourhood:
    1. Get the vector between the sheep and the neighbour, intuitively this is already 'scaled' proportional to the distance between the sheep and the neighbour.
- Return the average of the vector distances.

5.1.5 Separation

The final rule that Reynolds [1987] used was that of separation. This prevents the sheep from getting too close together, such that unrealistic or impossible movements and overlaps take place. The separation force will be greatest when near to another sheep. The algorithm for separation is as follows:

- For each sheep in the local neighbourhood.
    1. Inversely scale the distance between the sheep and its neighbour.
- Return the average.

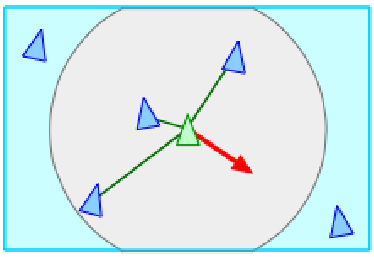


Figure 5: Separation [Reynolds, 1987]

5.1.6 Fleeing

The separation rule (fig. 5) prevents the sheep from becoming too close, in effect, counter-balancing the cohesion rule. In the same way, the effect of fleeing from a predator will be modelled in the same way.

- For each **sheepdog** in the local neighbourhood, apply the separation steering behaviour.

Since the flocking rules are reusable, the same algorithm can be used, parameterized with a different collection of animals (the sheepdogs) and a larger neighbourhood and weighting.

5.1.7 Randomness

All of the above rules work on the assumption that there are some other animals within the animals' neighbourhood. If there are no neighbours, the animal does not move. A random steering behaviour was used, which encourages movement when the animal is isolated. Instead of moving in a random direction each step, the animal will move in a particular direction for a random number of steps, to give a wandering style of movement.

5.1.8 Obstacle Avoidance

The environment was bounded by walls. The avoidance of the wall is modelled by a repulsive force. To make sure this force overrides any other forces, it is scaled by a massive value. No specific collision detection or avoidance of other animals is involved in the model, instead, the separation and flee steering behaviours are used to perform this task.

5.1.9 Combining the Rules

Once each of the rules has been calculated, they are combined to produce a single steering force that defines the acceleration of the sheep. In this most simple model, the rules are scaled and added together, before being normalised based on the maximum distance that a sheep can move in one time step.

```
Steering Force = a * Cohesion
                 + b * Alignment
                 + c * Separation
                 + d * Flee
                 + e * Randomness
                 + f * Obstacle Avoidance
```

Where a, b, c, d, e and f are constants that define by how much to scale each steering behaviour. The task of finding useful values for these constants was to a large extent trial and error, making small changes to the constants and running a simulation. The following ideas were used as a starting point:

- Cohesion, alignment and separation rules can be combined in a similar way to Reynolds' Boids [1987].
- The flee behaviour should be scaled by a significantly higher amount than the other flocking rules.
- The obstacle avoidance should be scaled by such a high value that it overrides any other behaviours, this prevents any unrealistic occurrences of sheep going 'through' walls.

5.1.10 Evaluation of Iteration 1

The initial sheepdog model did demonstrate a form of flocking behaviour which was a passable model for sheep. The rules were combined using a series of weightings which were chosen through trial and error, to be able to make them compete realistically. The sheep moved around and flocked, at a constant speed, they stayed within the bounds of the field, demonstrating that the wall avoidance behaviour was working. Once in a flock, they would align and continue moving at this speed. This was more similar to the behaviour of birds, it was clear that there needed to be some scaling on the speed of the sheep, based on their current situation.

In order to test the flee behaviour at this early stage in the development, a simple sheepdog agent was introduced to the simulation. The specifics of this agent are detailed in chapter 6, for the sake of this section, it is sufficient to know that the sheep should attempt to flee from the sheepdog, which should move in random directions across the environment.

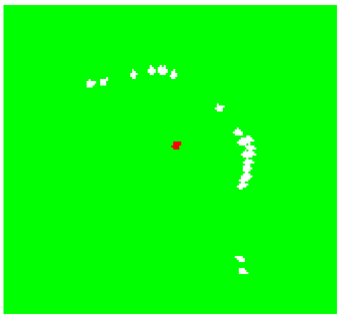


Figure 6: Sheep and a simple sheepdog model, demonstrating fleeing, flocking and collision issues.

The above screen shot (fig. 6) shows the flee behaviour exhibited by sheep in this initial model, the sheep are fleeing from the sheepdog in a reasonable fashion, while staying close to one another. In this figure, it appears that a collision has occurred. This is due to the limits of the animal's *agility* being reached, the separation force from other animals was not enough to avoid 'overlap' in the environment. Similarly, sheep formed tighter flocks when there were more sheep nearby. The minimum distance between sheep was bounded by the maximum amount of sheep that would fit into a neighbourhood, rather than how close sheep get to one another.

More comprehensive screen shots of the flocking behaviour and the interaction with predators are saved for the evaluation of the next iteration. The key reason for evaluating the model at this early stage is to illustrate the fundamental difficulties in achieving realistic sheep flocking behaviour using only Reynolds' [1987] flocking rules and a simple *flee* behaviour.

5.2 Sheep Model Iteration 2: A Layered Approach

In the previous iteration, another rule was added to the original flocking rules to avoid obstacles, or more specifically, walls. The fact that the weighting of this rule was large enough to dominate all other rules suggests that it did not belong in the flocking rules, since they were intended to be used as a combination of various competing behaviours. As well as this, there was no collision avoidance between the sheep themselves. These two issues prompted the need for a dedicated collision avoidance subsystem, separated from the flocking rules.

An element of sheep behaviour which was not addressed in the first iteration is the desire to simply wander and graze when not flocking or fleeing. This iteration develops a layered system of behaviours that incorporates them all. Similarly, the speed of the sheep when in a flock was far too high, since the sheep never appeared to form 'tight' flocks with little internal movement. The speed of the sheep was the same whether it was fleeing from a predator or just flocking.

A further improvement to be made was to modify each flocking rule so that it returned a normalised vector, of magnitude between 0 and 1. This was so that the weightings of the rules were not so arbitrary, some expectation could be made on the output of each flocking rule, making it easier to design different flocking characteristics, and potentially to create sets of weightings to represent various sheep personalities.

5.2.1 Speed

An improvement to the locomotion of the sheep was added, which varied the speed of the sheep depending on their particular state.

State	Percentage of Maximum Speed
Flocking	12.5%
Wandering	25%
Fleeing	100%

Figure 7: Sheep speed depending on state

This represents the sheep's desire to stay within a flock, by slowing down the sheep when it is with other sheep, and by speeding it up when it is alone. Additionally, if the sheep senses a predator within range, its speed will be increased to maximum. The steering behaviours now are then used to simply decide the direction of the steering force, before being normalised to the desired speed.

5.2.2 Normalising Steering Behaviours

In order to get sheep to behave realistically, a series of weightings were applied to the original flocking rules. The weightings used were arbitrary numbers and therefore could not be experimented with in any meaningful way. To deal with this, the steering behaviours were redesigned such that their values ranged between 0.0 and 1.0.

A further change to investigate was that of using different neighbourhood sizes for different steering behaviours. This allowed different steering behaviours to become active at different distances. The neighbourhoods were altered in the following ways:

- The flee neighbourhood was made larger than the other neighbourhoods.
- The cohesion neighbourhood was made larger than the separation and alignment neighbourhoods, to encourage the sheep agents to be attracted to each other from larger distances.
- The separation neighbourhood was reduced, since sheep do not have to consider the need to have enough space in order to fly, as in Reynolds' Boids model [1987].

The exact values for these neighbourhoods were experimented with, to find a model which seemed reasonable. The overall behaviour that was achieved is discussed in the evaluation of this iteration.

5.2.3 Layered Behaviours

In the first iteration of the model, the flocking, fleeing and wall avoidance behaviours were competing. This meant that it was possible for the separation force required to keep the animals far apart to be cancelled out by cohesion and alignment forces. In order to maintain the realism of the model, sheep cannot overlap or collide without a realistic reaction. A hierarchy of behaviours can be used to differentiate between the most and least important behaviours. This idea is based on the work by Brooks [1986], which uses a horizontally layered architecture to perform action selection for robots.

Layer	**Behaviour**
1	Avoid collisions
2	If predator near, apply flocking rules
3	If hungry, graze
4	If 4 nearby neighbours, apply flocking rules
5	Wander

Figure 8: Layered behaviours for sheep

The lower level behaviours inhibit the higher level ones. Therefore, if there are other animals in the *collision zone*, the first layer will take precedence over all the others. If there are no animals in the *collision zone* then the next layer is evaluated. Since flocking only occurs in groups of four or more sheep [Weaver, 2005], it is not always necessary for the flocking layer to be used. If there is a predator within the *flight zone*, then flocking will take place, since it is primarily a form of defence [Gill, 2005]. If there is no predator nearby, the third layer is activated, which simulates grazing if the sheep is hungry. This behaviour was not implemented in this iteration, however, the way in which the layered hierarchy works allows it to be in place ready for when the emotion subsystem is implemented. The fourth layer performs flocking if there are enough nearby sheep. Finally, if there are no nearby sheep or predators, the sheep wander randomly around the field.

The actual collision avoidance layer is simple, it uses a single separation steering behaviour, with a very small neighbourhood, known as the *collision zone*, and a very large weighting.

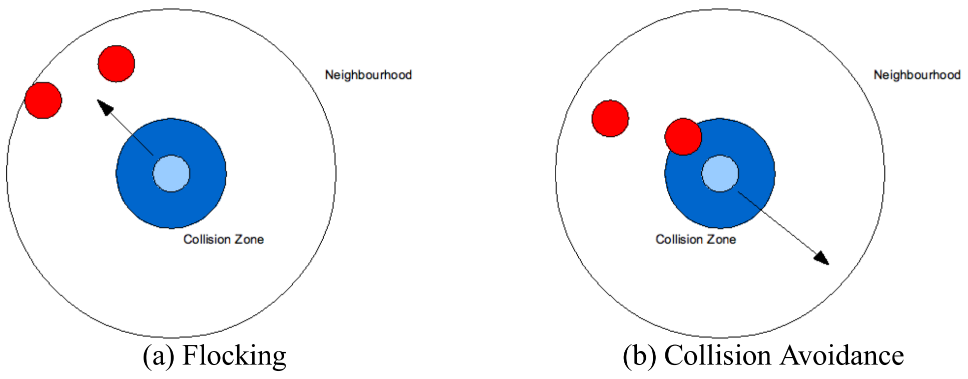


Figure 9: Activating the collision avoidance layer.

In the above figure (9 (a)) we see that flocking occurs normally when the collision zone is clear. In this case, the normal flocking rules are still active, including the 'weak' separation force. However, in (b) we see that an animal has entered the collision zone, causing Layer 1 to activate, producing a strong separation force from the animal in the collision zone.

Due to the way that the flocking rules are implemented, it is possible for layer 4 to produce very small movements, such that it is almost impossible to see any movement at all. For example, when there are very few sheep, on the cusp of a sheep's neighbourhood, the magnitude of the steering behaviours may be very small indeed. A simple solution to this was to add a threshold to the magnitude of the steering behaviour in layer 4. If it is below a certain small value, for example 0.01, then the layer will be bypassed, moving on to layer 5.

5.2.4 Evaluation of Iteration 2

The layered hierarchy worked well, allowing various behaviours to be arranged into a particular order of precedence. It was a useful addition since it made extending the system for emotion easier, the next iteration will be able to plug behaviours into the appropriate layer. It also meant that the flocking rules could be used as originally intended, within their own layer, without having to compete with other forces such as obstacle avoidance. Similarly, the improvement to the way in which the steering behaviours were normalised and combined, as well as the neighbourhood for each behaviour, did improve the flocking behaviour of the sheep.

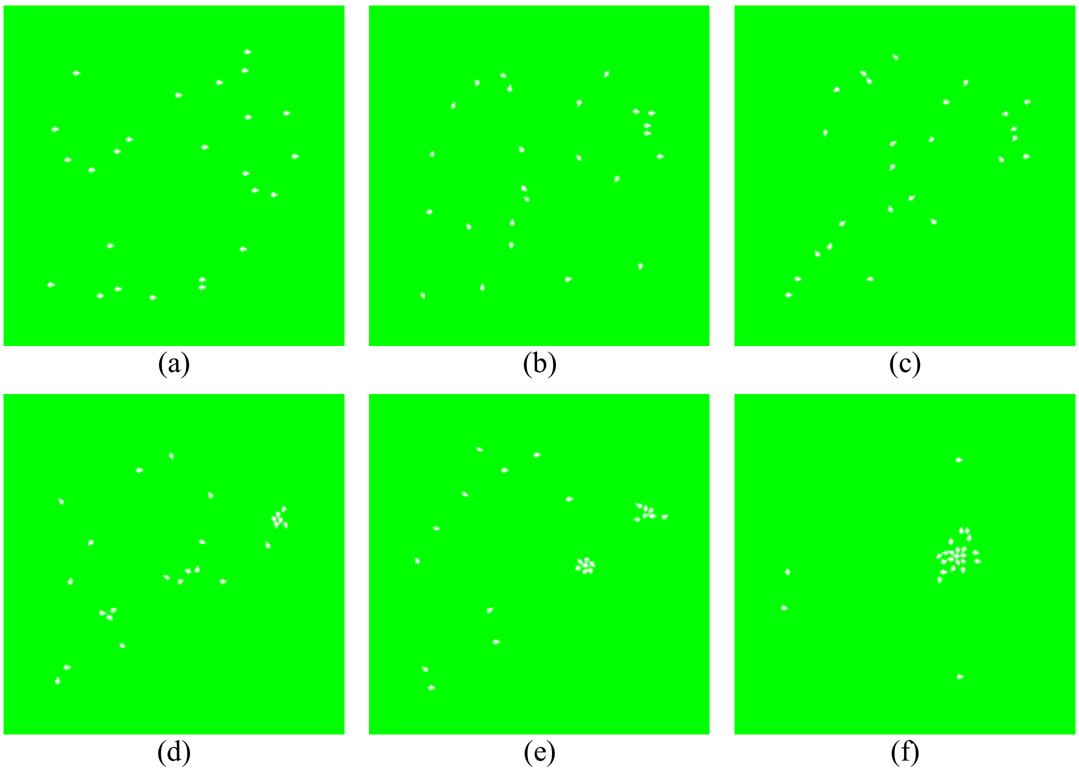


Figure 10: Flocking behaviour using a layered architecture.

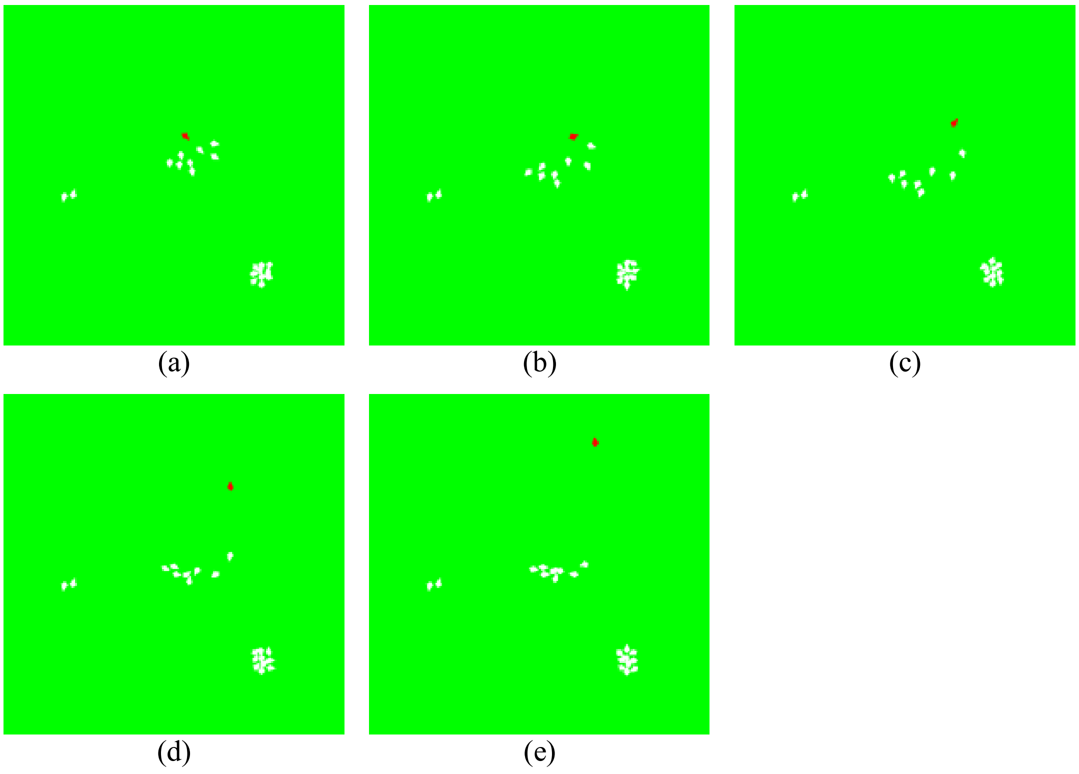


Figure 11: Flocking behaviour with a single predator

Figure 10 shows the progression of the flock. Over time the sheep begin to form flocks with their neighbours, until the final screen shot (fig. 10 (f)) shows an almost complete flock. If the simulation is left to run for long enough, it is probable that all sheep will join a single flock. No collisions appear to happen, the collision avoidance layer has enforced a minimum separation distance between sheep. In both (fig. 10 (e)) and (fig. 10 (f)), we can see that regardless of the size of the flock, there is always at least some separation between sheep. Even with no predator, the sheep will wander around the field (fig. 10 (b)) and tend towards flocks, with some sheep near to a flock but not amongst it, fulfilling their desire to graze. What is difficult to tell from the screen shots is the speed at which the sheep are moving within flocks, however it was observed that the movement within a flock was at a more appropriate pace.

Figure 11 shows how a randomly moving sheepdog, as described in iteration 1 was added to the simulation. We see both the flocking of the sheep which are away from the predator, as well as the fleeing from the predator, without collisions (b). Overall the model at this stage was satisfactory, it was enough to build on when implementing sheep emotions and personalities, and would have been good enough to implement gathering and herding behaviours for the sheepdogs.

5.3 Sheep Model Iteration 3: Sheep Emotion and Personalities

Iterations 1 and 2 were concerned with creating a realistic sheep flocking model, these iterations focused on using a homogeneous, reactive sheep model, that is, every sheep followed the same rules. The third iteration of the sheep model looks at modelling the emotions of a sheep, as well as the different personalities that are exhibited. With the normalised values for the various steering behaviours from iteration 2, it is possible to modify the flocking behaviour, using weighting or neighbourhood values to create different behaviours. An updated flow diagram for the sheep agent is as follows:

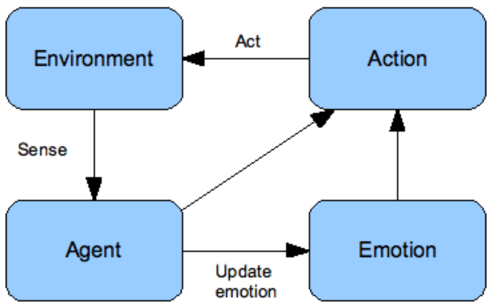


Figure 12: Reactive agent with emotion subsystem

Due to the time constraints of this project, we focussed on three 'emotional' factors, hunger, fear and leadership.

5.3.1 Hunger

Hunger was used as a way to differentiate between the sheep's desire to flock and need to graze. On each step, the sheep will become *hungry* with a probability *p*.

```
func step()
        if random boolean (p)
                        hungry = true;
        endif
end func
```

Once hungry, the sheep will attempt to stand alone and still, the layered architecture (fig. 8, p40) shows that this can only happen when there are no immediate collisions and no predators nearby. Once doing this, the sheep will return to the 'not hungry' state with probability *q*.

```
layeredBehaviours()
if(...layers 1 & 2 of heirarchy ...)
... ...
else if(hungry)
        movement = (0.0, 0.0)
        if random boolean (q)
                        hungry = false;
        endif
else if (...)
... (rest of layered hierachy) ...
end func
```

A values of 0.001 was chosen for p and q, meaning that when undisturbed by predators or collisions, a sheep will change between 'hungry' and 'not hungry', with probability 0.001.

5.3.2 Fear

The fear response is used as a way to model how much and how often a sheep should flee from predators. As discussed in the literature review, sheep and sheepdogs are believed to have a dynamic such that sheep have a certain level of respect for a particular dog [Downe, 2005]. Therefore, the fear response was based on the 'respect' value for nearby predators. A higher respect value *or* a higher number of predators will result in the fear emotion increasing more rapidly.

Each sheep has a value representing how much fear it feels at a particular time. When a predator comes within the sheep's flight zone, the value increments on each step of the simulation. How much the value increments by is based on:

- The leadership value of the sheep, that is, how brave it is.

- The aggression of the sheepdog.

Once the sheep's flight zone is clear of predators, its fear counter decrements.

```
int fear            // a counter, recording the animals fear level from one step to the next
func isFearing()
        if predators are near
                for each nearby predator p
                        increment fear depending on the respect level for p
                end for
        else
                decrement fear
        end if
        return fear > 0
end func
```

The fear emotion is updated using the algorithm above, and fits into layered architecture from (fig. 8, p40) as follows:

Layer 2	If predator near, apply flocking rules

becomes

Layer 2	If feeling fear, apply flocking rules

This allows for the possibility of sheep to feel the need to flock, even when the predator is no longer in range, since the layered model checks for the presence of a fear value, which persists even after predators are out of range, rather than checking for predators being near at a particular instance. In the same way, different sheep can have different *fear thresholds* that cause them to react to fear at different times. For example, a sheep with a high fear threshold would not begin to flee until a predator has been in range for a certain amount of time.

5.3.3 Leadership

A constant was added to each sheep, representing its leadership or dominance value. When considering nearby sheep for flocking rules, usually only location and momentum were taken into account. An extra factor was considered, the magnitude of steering force towards or away from another sheep depends on each sheep's leadership value. This is different to the way in which weightings of the steering behaviours are applied, since they are based on the average of all nearby sheep rather than the interaction between individuals.

In order to model the response to leadership, the generic sheep model takes into account how much *influence* a particular sheep can have on itself.

- Sheep are more likely to align with a sheep with high leadership.
- Sheep are more likely to cohere with a sheep with high leadership.

As well as these modifications, the leadership value of a sheep affects its own weightings for the flocking rules. A sheep with high leadership:

- Does not align as strongly with other sheep.
- Does not cohere as strongly with other sheep.
- Separates with a greater magnitude over other sheep.

In the same way, a sheep with a less than average leadership value exhibits the opposite of the above rules.

5.3.4 Sheep Personalities

The original sheep model which was developed in the first two iterations was known as the 'standard' sheep personality. A further four personalities were implemented: dominant sheep, outlier sheep, heavy sheep and light sheep. The following section discusses the key issues in modelling each personality, explaining how the flocking rules were modified and used in conjunction with the emotions described above.

Dominant Sheep

The most basic modification to the sheep model, is the introduction of a sheep that exerts greater influence over other sheep. This sheep, sometimes known as a Bellwether [Sheep, Animal Corner, 2008] is able to lead the flock. Intuitively, the dominant sheep had an increased leadership value as described above, causing other sheep to be more attracted to it, modelling the 'follow-the-leader' behaviour exhibited by sheep [Budiansky, 1999]. A less obvious change is the increased separation and reduced cohesion and alignment to other sheep. This causes a dominant sheep to give the impression of leading the flock, since it tends towards the edge of the flock, beginning to separate from the flock. Combined with the leadership value, other sheep will be inclined to follow. This is far simpler and is more in keeping with the traditional flocking approach, rather than implementing explicit 'follow-the-leader behaviour, where there are scripted techniques and formations for following a leader [Reynolds, 1999].

Outlier Sheep

An outlier [Sheep, Animal Corner, 2008] is a sheep which prefers to flock less, and instead be near the edge of the flock with few close neighbours. This may be due to the lack of strength of the sheep to push to the centre of the flock or the inability to graze when in a large flock.

The outlier sheep was modelled in a similar way to the dominant sheep, it has a higher separation and lower cohesion weighting, causing it to linger on the edge of the flock. However, its *reduced* leadership value discourages other sheep from following it, preventing it from steering the flock with its outlier tendencies.

Heavy Sheep

As described in by Downe [2005], a heavy sheep is very stubborn but highly cohesive. This sheep was implemented as a means of testing out the effectiveness of sheepdogs when herding sheep that appear to stand up to them. The main characteristics of the heavy sheep was an increased cohesion weighting, and a lower separation value to encourage tighter flocks. The neighbourhood in which the heavy sheep responds to predators is smaller than that of the standard sheep, modelling its reluctance to flee from predators, similarly the fear threshold, that is, the amount of fear required before fleeing will even take place is higher.

Light Sheep

The light sheep was the opposite of the heavy sheep in most respects, it was used to test the effectiveness of a sheepdog for gathering or herding particularly nervous or hard to control sheep. The light sheep is slightly less cohesive and has a larger flight zone, or flee neighbourhood, making it flee from predators more quickly than other sheep.

Overview of Sheep Personalities

The following table (fig. 13) shows how the flocking rules and scaling factors were changed for each sheep personality. The solution was developed iteratively, therefore, arbitrary scaling factors were unable to be *designed*. The table summarises which weightings increased, decreased or stayed the same, in relation to the standard sheep personality.

Personality	Cohesion Weighting	Separation Weighting	Alignment Weighting	Flee Neighbourhood	Leadership	Fear Threshold
Dominant	-	+	-		+	
Outlier	-	+			-	
Heavy	+	-		-		+
Light	-	+		+		

Figure 13: Relative scaling of flocking rules and personality constants for different sheep personalities.

A + represents an increase relative to the standard sheep personality, while a – represents the opposite. Hunger is not included in the table since it is not varied for different sheep personalities, it is used in the same way, regardless of personality.

5.3.5 Evaluation of iteration 3

This iteration built upon a stable sheep model, in order to add variation in the form of emotion and personality. The following screen shots provide a feel for how the different personalities affected the simulation.

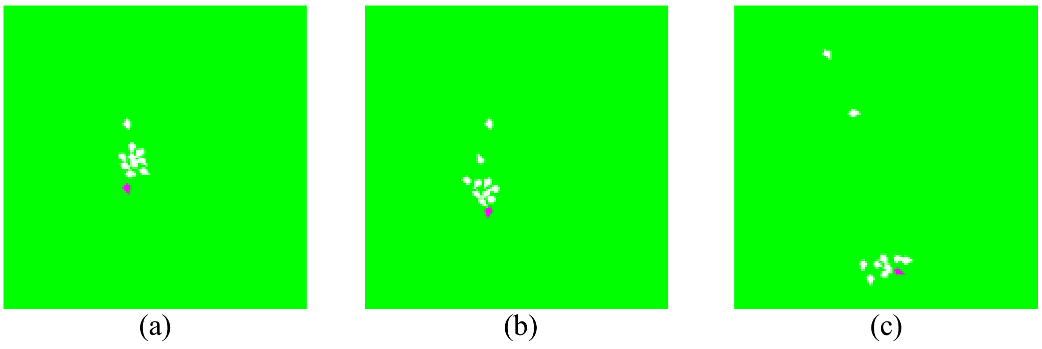


Figure 14: The dominant sheep (pink)

In the above figure 14, we can clearly see the flock being guided by the dominant sheep (pink). Some sheep have been left behind, since they were grazing as the flock moved away, and there is no predator nearby to cause them to flee or search for a flock.

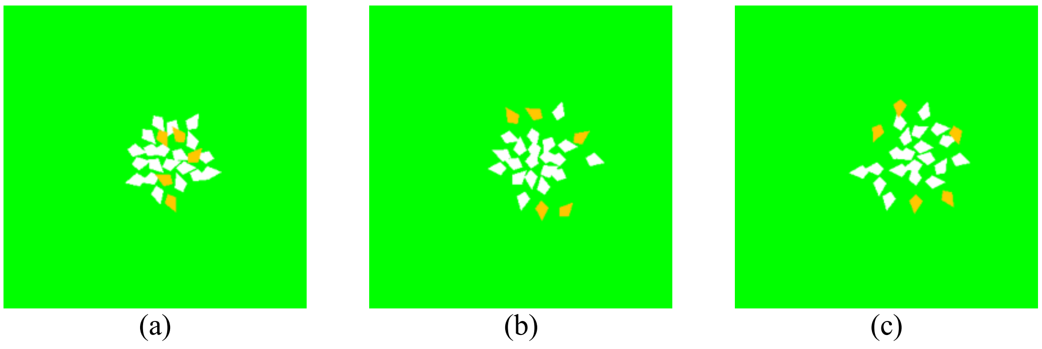


Figure 15: Outlier sheep (orange)

The outlier sheep was added to the simulation in the middle of the flock. The above screen shots (fig. 15) show that as the simulation progresses from (a) to (c) , the outlier sheep make their way to the edge of the flock and stay there, while the rest of the sheep compete for space at the centre of the flock.

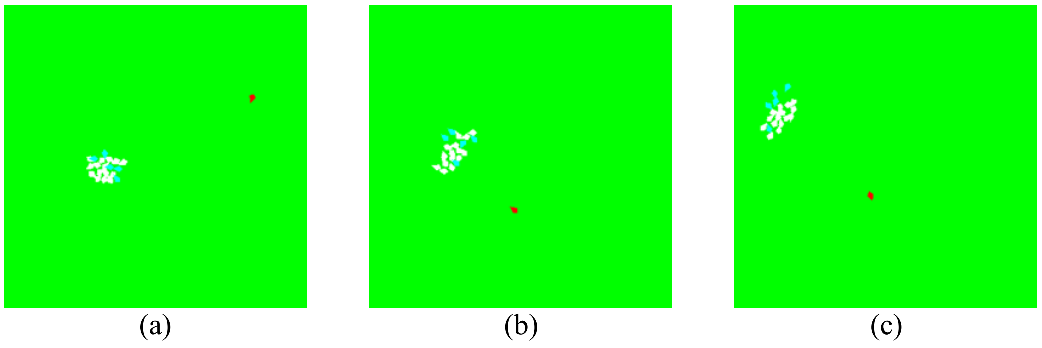


Figure 16: Light sheep (cyan)

A more subtle variation was that of the light sheep. Figure 16 shows that when a predator (red) comes near, the light sheep flee more urgently and end up furthest away from the predator.

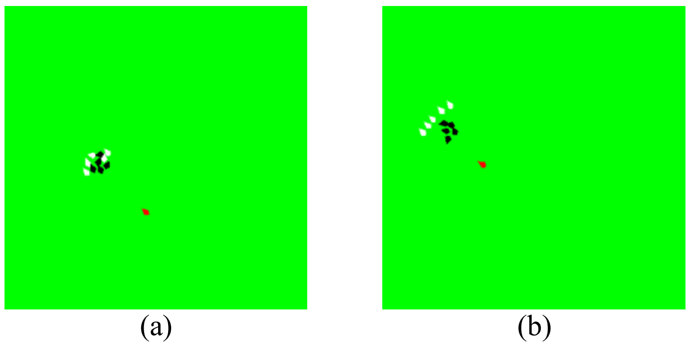

(a) (b)

Figure 17: Heavy sheep (black)

Finally, the heavy sheep (fig. 17) represents a stubborn sheep. We can see from the above screen shots that these heavy sheep (black) are less likely to flee from the predator (red), allowing it to get closer to them than the standard sheep do.

Overall we have seen a variety of sheep personalities in an isolated environment. In order to fully test out the range of behaviours, more in depth testing is detailed in Chapter 8, highlighting the differences in gathering and herding various combinations of sheep.

Chapter 6, Sheepdog Model Design

Several iterations of sheepdog design were performed, aimed at developing sheepdogs that could perform different tasks such as gathering and herding.

6.1 Sheepdog Iteration 1: Basic Sheepdog Model

In order to test the reactions of sheep to predators, a simple sheepdog was used. This allowed the implementation of the sheep to take into account predator reactions before the sheepdogs were fully implemented. This basic model involved random movement of the sheepdog, within the bounds of the environment, in order to see the flocks reactions to a predator (fig. 18). The random wandering of the sheepdog exerts repulsive forces on the sheep, causing them to exhibit their fleeing behaviour.

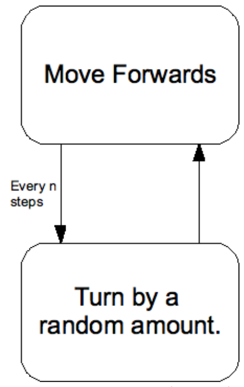


Figure 18: Basic sheepdog flow diagram

6.2 Sheepdog Agent Architecture

Aside from the basic sheepdog described above, sheepdog agents were implemented as practical reasoning agents [Wooldridge, 2002]. This consisted of them having a goal to achieve, be it to gather the sheep into a flock, or to herd the flock to a particular location. The main issues to consider when implementing such an agent are:

- What is the goal?
- How will the goal be achieved?
- When do we stop trying to achieve the goal?
- How do we know when the goal has been achieved?

The project focuses on two main types of task, gathering sheep and herding sheep. Two main sheepdog agents are implemented to achieve each of these tasks. The following sections discuss how each agent addresses the issues described above.

6.3 Sheepdog Iteration 2: Gathering Sheepdog Model

The main goal of the gathering sheepdog is to create an agent which gathers all of the sheep in the environment into a single flock. The sheepdogs attempt to disturb the sheep as little as possible, by using a form of obstacle avoidance when navigating around the flock. The gathering sheepdog agent was intended to be able to work with other sheepdogs without any external coordination, such that the speed of the gathering task is increased.

6.3.1 Gathering Techniques

In order to gather sheep into a single flock, the sheepdog needed a generic way of achieving the goal. Several methods were considered which would allow the sheepdog to generate a plan for gathering each sheep, stating where to gather the sheep to and the order in which to gather them. The following techniques were implemented:

1) Naïve gathering method,

1. Find the overall centre of flock, by averaging the positions of the sheep.
2. Find sheep furthest from the centre of the flock, gather it to centre.
3. Move to next sheep (step 2) when the sheep is within a certain tolerance range of the centre of the flock.
4. If all sheep are within range of the centre of the flock, but are not in a tight enough flock, repeat with a lower tolerance 'range'.

2) Goal Directed

This involved working with a specific goal, and worked as method 1 did, gathering towards a specific location as opposed to the average flock centre.

6.3.2 Steering Points and Pivot Points

The fundamental idea behind shepherding is the use of repulsive forces to move another being in a particular direction. In this case, the sheepdog exerts a repulsive force on the sheep, because the sheep attempt to flee directly away from the sheepdog. Lien et al., [2004] identify the concept of a steering point as the point from which the sheepdog can push the flock towards its goal. We introduce a modification to this idea, that is, a steering point as well as a *pivot point*.

Both the steering point and the pivot point lie on a line that passes directly through the target location and the sheep to be moved (fig. 19). The steering point is placed at a suitable distance away from the sheep such that when the sheep dog is at the steering point, the sheep are not attempting to flee. The pivot point is placed within the sheep's flight zone. Once the sheepdog has reached the steering point, it attempts to reach the pivot point, it may never reach it, depending on the flight zone of the sheep and their sensitivity to the sheepdog.

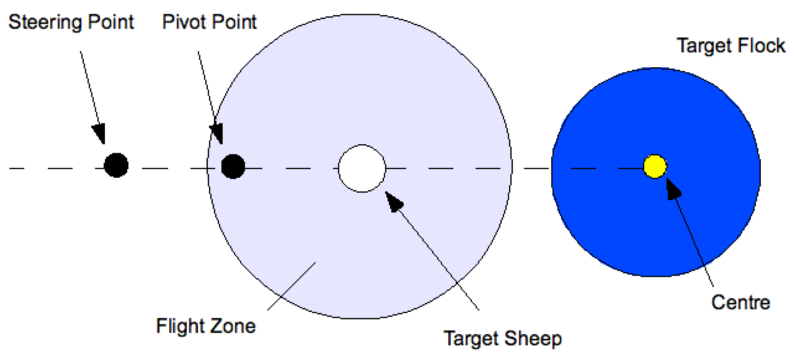


Figure 19: Steering Points and Pivot Points

The addition of the pivot point enables the dog to get behind the sheep without actually being inside the sheep's flight zone. This idea allows for the possibility that the sheep may stand up to the sheepdog. By moving to the pivot point and waiting, rather than going straight into the sheep, the sheepdog increases the sheep's fear emotion, equivalent to that of using 'eye' [Downe, 2005]. Once the sheep is feeling enough fear to flee, it will move. On each step of the simulation, the steering point and steering pivot are recalculated, thus making the pivot point move and causing the sheepdog to pursue the sheep to the goal.

6.3.3 Objective Vector

Unlike the sheep model, a sheepdog has a goal to achieve, therefore we use the concept of an *objective vector*. This defines a particular location that the sheepdog wishes to move towards, be it a steering point, a sheep or any arbitrary location. The objective vector is combined with any other competing behaviours such as obstacle avoidance, to produce the sheepdog's overall movement vector.

6.3.4 Approaching Issues

In order to move a flock or single sheep in a particular direction, the sheepdog must get behind the sheep and apply a repulsive force towards the goal as shown in section 6.3.2. Two different approaches were implemented:

1) **Straight Line approaching.** This simply means moving straight towards the steering point, regardless of what might be in the way.

2) **Safe-zone approaching.** This works by approaching the steering point in as straight a line as possible, while avoiding any sheep along the way, to avoid breaking up any existing flocks.

Implementing the first approaching method is as straightforward as calculating the steering point, based on: the location of the target, the current position of the sheepdog and the desired steering distance (fig. 19). The simple trigonometry used for this calculation can be seen in the source code, *SheepDog.java*, on the attached CD. To implement the second method we used a more advanced technique; a form of obstacle avoidance was implemented.

6.3.5 Obstacle Avoidance

The fundamental difference between a separation steering behaviour and obstacle avoidance is the requirement for the agent to *reach a goal* whilst also avoiding others. In the case of separation, the goal, or objective is to move away from a particular agent, whereas obstacle avoidance the goal is to move towards some *other* objective while avoiding particular obstacles.

It can be shown with the aid of a simple diagram why a separation steering behaviour is unsuitable for achieving obstacle avoidance, even when combined with an objective vector.

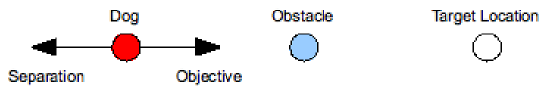


Figure 20: Obstacle avoidance with flocking rules

In the above diagram (fig. 20), the dog is attempting to reach the target location, whilst avoiding an obstacle; all three points are on a straight line. The dog combines two steering behaviours: the objective of moving towards the target location and the separation moving away from the obstacle. It is evident that this combination of forces will result in a deadlock, therefore a more sophisticated technique was required.

Lien et al. [2004] identify the idea of safe-zone approaching. A circle of influence is drawn around flocks of sheep, and the shepherd avoids these zones when approaching the steering point. For this project we instead decided to use something similar to Reynolds' [1999] obstacle avoidance steering behaviour. This method has the advantage of being applicable to objects occupying a single point in a 2D environment, whereas the concept from Lien et al. [2004] assumes the existence of flocks, occupying a polygon within the environment, as well as the ability to navigate round arbitrary sized shapes. Reynolds' [1999] style of obstacle avoidance allows the sheepdog to avoid sheep without any knowledge of flocks. Using the sheepdogs momentum, it is possible to work out which way the sheepdog is facing. From this we draw a bounding box in front of the sheepdog, representing the area in which the sheepdog will attempt to avoid obstacles in. The bounding box is split into a left and right section.

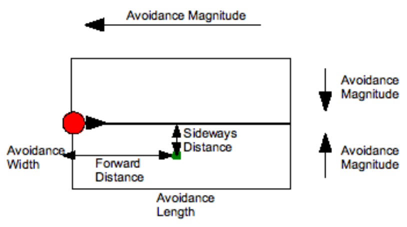


Figure 21: Obstacle avoidance area

For each box, the sheep contained within that box are found, and the magnitude of the steering force required to avoid the sheep is calculated.

The pseudo code for calculating the magnitude required to avoid sheep in a particular box is as follows:

```
func Force findMagnitudeForSheep( sheepInBox ):
Force overallSteeringForce;
For each Sheep in sheepInBox
    overallSteeringForce += (a * (Avoidance Length – Forward Distance to sheep)
                             + b * (Avoidance Width – Sideways Distance to sheep));
end for
return overallSteeringForce;
end func
```

As shown in figure 21, the steering force required to avoid each sheep is inversely proportional to the 'forward' distance and 'sideways' distance from the sheepdog. The constants a and b are used to scale the importance of the object being close to the sheepdog, or the object being near to the sheepdogs line of movement. Various values were experimented with, values of around a=0.75 and b=0.25 seemed to work most effectively.

The steering forces for each box were compared using the following simple algorithm:

```
Find the sheep in the left and right boxes of length Avoidance Length and width
Avoidance Width.
magnitudeLeft = findMagnitudeForSheep( sheep in left box )
magnitudeRight = findMagnitudeForSheep( sheep in right box )
AvoidanceForce = Maximum(magnitueLeft, magnitudeRight);
```

This takes each side of the avoidance box and calculates the magnitude required to avoid the sheep on each side. The side with the greatest magnitude is chosen as the steering force, instead of taking the sum of the two vectors. If we used the sum of the vectors, it would be possible to get into a deadlock:

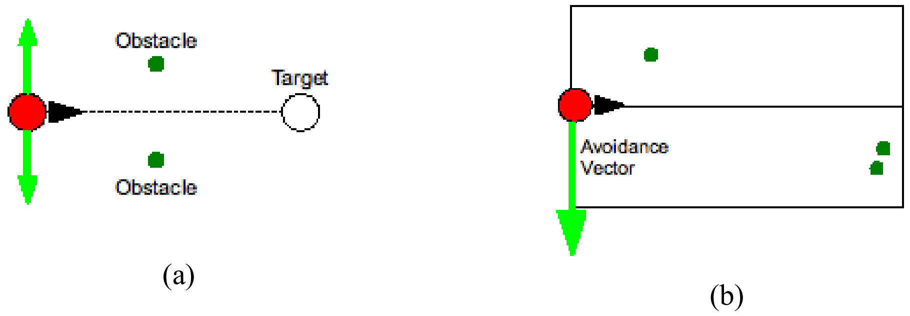


Figure 22: Obstacle avoidance issues

Here, (fig. 22(a)) there are two obstacles, equal distances from the sheepdog. The magnitude of the avoidance vector for each is equal and opposite, giving a zero vector when added, causing the sheepdog to move straight towards the objective without avoiding the obstacles. To avoid this we always choose the largest steering force, or if the forces are equal, pick an arbitrary direction and maintain this avoidance direction for at least 50 steps of the simulation, to prevent the avoidance from going back and forth each step.

Figure 22(b) shows that even though there are more sheep in the right-hand box, the one sheep in the left-hand box is much closer, causing it to have a higher avoidance magnitude. It is important to note that the sheepdog cannot totally avoid sheep, otherwise progress towards the objective may never be made, especially in a busy field. Instead, the avoidance gets a particular weighting with the overall *objective vector*, giving the obstacle avoidance a 'best as possible' approach. This behaviour was implemented such that it could be used by various implementations of the sheepdog, including both the gathering dog and the herding dog as described later.

6.3.6 State Machine

It has been shown that the hunting behaviour of lions can be modelled by applying the three basic flocking rules [Barry and Dalrymple-Smith, 2005]. In terms of sheepdogs, it was decided that a state based approach would reduce the complexity of combining these rules at particular times. Defining a state transition function based on particular conditions allows the sheepdog to operate in various different states. Depending on the state, different rules are executed with different size neighbourhoods and different weightings.

The following diagram illustrates the state transitions for a gathering dog.

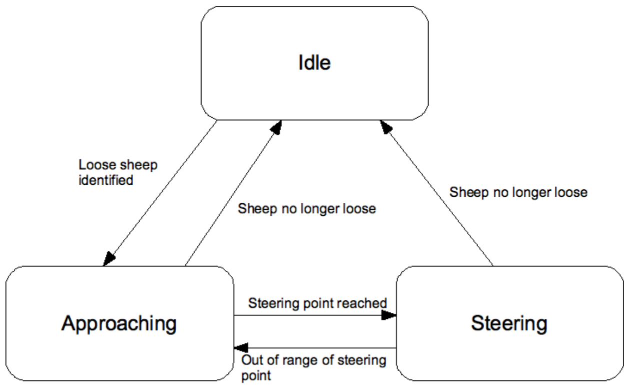


Figure 23: State transition diagram for a gathering dog

The following table represents the state transitions in a more formal way, making it more appropriate for implementing.

State	Sense	New State
Idle	Loose Sheep	Approaching
Approaching	Target no longer loose	Idle
Approaching	Reached steering point	Steering
Steering	Target no longer loose	Idle
Steering	Went out of range of steering point	Approaching

Figure 24: State transition table for a gathering dog

Hysteresis

When changing between states, a hysteresis was used to prevent constant switching back and forth between states when on boundary input conditions. For example, when going from approaching to steering the following state transition occurs:

```
if Dog is within range X of steering point, change state to Steering
```

Once in the steering state, the fallback condition for when the dog goes out range of the steering point is as follows:

```
If Dog is NOT within range (X + c) of steering point, change state to Approaching
```

A tolerance of c is added when changing back to the approaching state. This means that knowledge of the dogs proximity to the steering point is not enough to decide which state the dog is in, the current state must also be known.

6.3.7 State Based Movement

The states described above are used to modify the sheepdog's behaviour, by changing the objective vector, using obstacle avoidance or by modifying the weightings of the flocking rules. A summary of each states properties is as follows:

Idle State

Increased **cohesion** value to the sheep, so that the sheepdog keeps the flock tight and prevents any difficult sheep from separating from the flock, also known as *covering* the flock. [Lien et al., 2004]

Increased **separation** from other dogs, so that maximum covering of the flock takes place. This also gives multiple sheepdogs a lower chance of moving towards the same sheep when there are still sheep to be gathered, since their local view of the sheep flocks will be different.

Approaching State

The **objective** of the sheepdog is to move towards the steering point for the sheep it is approaching.

Perform **obstacle avoidance** on sheep, so that the flock is not disturbed while approaching the steering point

Steering State

The **objective** of the sheepdog is to move towards the pivot point.

For each state, the various vectors are combined to produce the sheepdog's overall movement vector.

6.3.8 Multiple Gathering Dogs

The gathering sheepdog was intended to work on a single sheep at a time. Therefore, the only cooperation that was beneficial to these dogs was to ensure that they worked on different sheep. Since it was decided that there would be no external coordination of the sheepdogs, the sheepdogs must be able to act based upon their own perception of the environment, rather than being instructed which sheep to gather.

This was achieved by modifying the behaviour of the gathering dog when it is in its 'idle' state, that is, when it is deciding which sheep to herd next. The sheepdog will attempt to avoid other dogs while it decides which sheep to herd. This involves making a modification to the algorithm used to decide which sheep to gather. Instead of looking at the field as a whole and finding the farthest sheep, the sheepdog considers sheep within its own neighbourhood, and targets the farthest sheep from this subset of the field.

6.3.9 Gathering Sheepdog Evaluation

The gathering sheepdog represented an agent which was successful in gathering a field of sheep into a single flock. The idea of using a steering point to move a sheep using repulsive forces was extended to use both a steering point and steering pivot, this worked well, allowing the sheepdog to approach the sheep with a wide berth, without unnecessarily disturbing it.

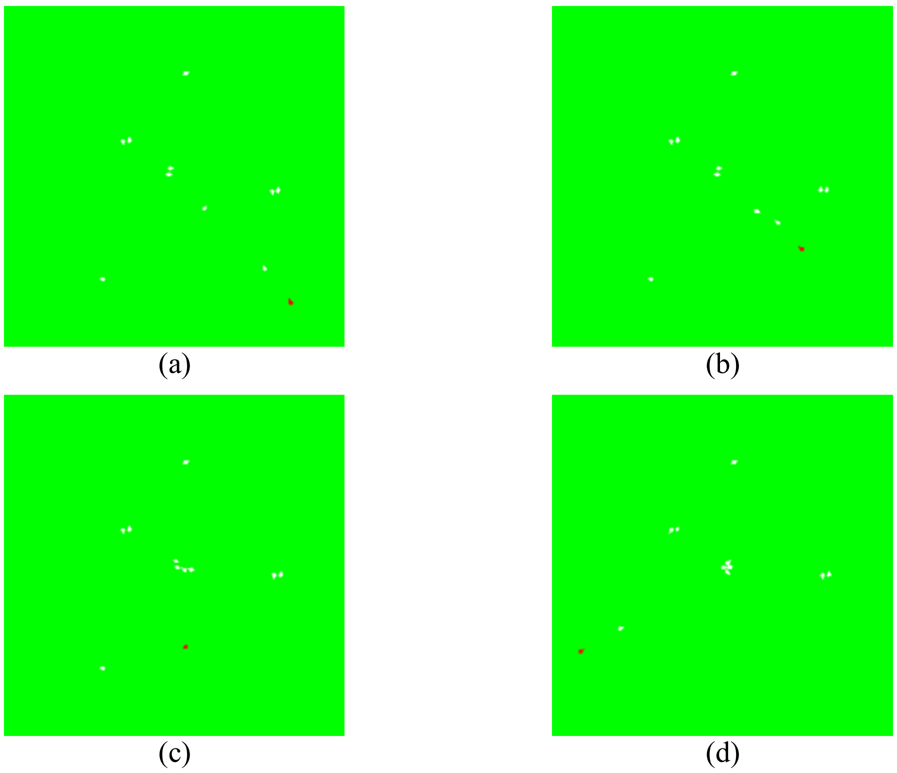


Figure 25: Single gathering sheepdog.

Figure 25 shows the sheepdog (red) herding a single sheep (a), (b), before moving on to its next target (c) (d). The sheepdog has approached the sheep using safe-zone approaching, and has not disturbed any other sheep along the way.

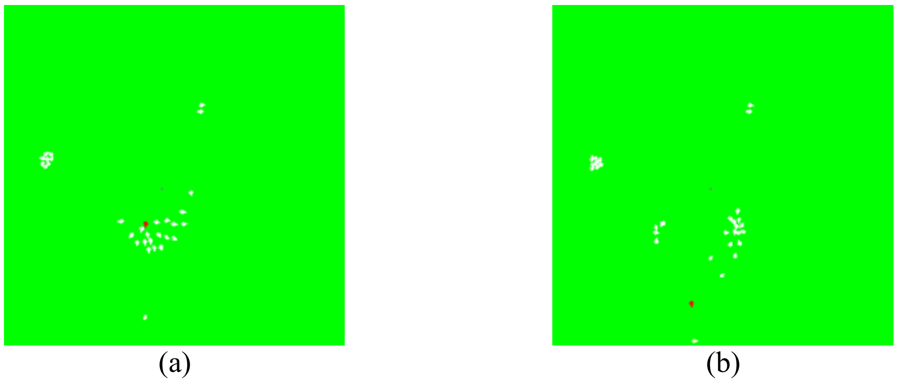


Figure 26: Gathering sheepdog with no obstacle avoidance

The above screen shots (fig. 26) demonstrate the issues faced when no obstacle avoidance was used, the sheepdog has disturbed the flock as it approaches the sheep at the bottom of the field.

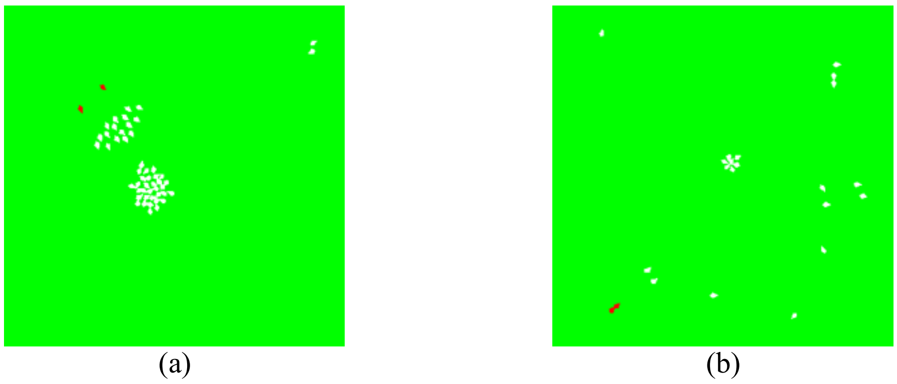


Figure 27: Emergent behaviour through cooperation

The above screen shots (fig 27) show two cases of emergent behaviour, one positive and one negative. In (a) we see two sheepdogs seemingly working together, even though there is no cooperative behaviour implemented when steering sheep. This has happened because the two dogs are targeting different sheep which happen to fall in the same flock, causing the two sheepdogs to appear as if they are herding the flock as a whole. In (b) we see a less useful interaction, both sheepdogs are overlapping as they attempt to gather the same sheep, diminishing any benefit gained from multiple sheepdogs.

It was clear from this sheepdog model that the sheepdogs needed to be able to work on flocks as a whole rather than single sheep, so that they can cooperate to drive groups of sheep more effectively. This required some cooperation when steering the sheep as well as deciding which sheep to herd.

6.4 Flock Identification

So far, the design has focused on gathering single sheep, one at a time, into a flock. A more interesting shepherding behaviour is that of herding [Lien et al., 2004], in which the sheepdog works on a group of sheep collectively. In order to be able to implement this, some concept of a flock had to be identified, allowing the dog to recognise the size and shape of the flock it is dealing with. Once a flock has been identified, it is possible to analyse the overall momentum and alignment of it and use this information to steer the flock.

Initially, an algorithm was implemented which would identify flocks on every step of the simulation, drawing bounding boxes around groups of sheep which are within a predefined distance of each other. This worked to an extent, however there was no concept of maintaining a flock from one step to the next. The sheepdog had to rely on its own best guess to work out where its current target flock had moved to on each step. To make it possible for a sheepdog to maintain its goals, it was necessary to have some way of identifying flocks from one step of the simulation to the next. Since the model was not designed to implement the true vision of a sheepdog, it was necessary to have a way of uniquely identifying a flock. A solution was implemented which would allow the sheepdog agents to maintain a reference to a flock object, which would persist across simulation steps.

6.4.1 Centralised versus Distributed Flock Recognition

Considering the possibility of multiple dogs working together, a choice had to be made about how to organise the flock recognition, using one of the following methods:

1. **Each dog has a local view of its environment, and identifies flocks using this information**

   This has the benefit of giving each dog a personalised view of the flock, more closely representing how they may perceive the flock in the real world. When cooperating this would make it similar to the cooperative gathering dogs described earlier. Instead of needing to specify some way of deciding which dog goes where, the dogs could simple separate from each other then act on their local environment.

2. **Each dog has the same, complete view of its environment.**

   This approach would be simpler, since it requires only one centralised computation of the flocks. It does introduce the issue of having some additional mechanism to allow cooperation to occur, although once computed, it is simple for each dog to use the single source of information to calculate which flocks are nearby.

It was decided that the most effective method of achieving this would be to use method 2, and to provide a single, global view of the flocks in the field, which would be available to all dogs.

6.4.2 Flock Representation

When talking about flocks of sheep, it is necessary to define some bounds at which the sheep is either considered *in* or *out* of the flock. Intuitively, it is necessary to define where the flocks are, before a sheep can be considered *in* or *out* of it. The possible considerations for this were:

- A sheep with a particular number of neighbours is in a flock.
- A sheep is in a flock if it is within a certain distance of any other sheep in the flock.

The second method was used for this project. Using this idea, the concept of the flock is transient, and arises out of the proximity of sheep, it is not an entity within its own right. This gives rise to the issue of how to identify a flock from one step of the simulation to the next. There are several scenarios to consider when attempting to identify flocks over multiple steps.

- A sheep leaves a flock.
- A sheep joins a flock.
- A flock separates into *n* other flocks.
- *n* flocks join together.

For a sheepdog to be able to work with a particular flock from one step to the next, the identification system had to cope with all of these scenarios in a predictable and consistent manner.

A flock was represented by a bounding box, enclosing all sheep within the flock, including a small amount of padding. A flock was identified by the sheep that is closest to its centre. As the simulation moves from one step to the next, the new flock position is calculated by branching out from the sheep which was closest to the centre during the previous step. This gives the potential for the following special cases:

- Two flocks merge – In this case, the bigger flock will take in the smaller flock, the smaller flock no longer exists.
- A flock disbands – All sheep from a particular flock are now part of other flock(s).

These points had to be carefully considered when implementing the flock identification system.

6.4.3 Techniques

The following techniques were considered and helped to move towards the chosen technique:

Average of all sheep

The most simple method, this takes the positions of all sheep in the environment, and then averages them to produce an average flock position. It was obvious that this technique would generally be useless in terms of identifying where the flocks are, as more often than not an average position would not be near to any sheep at all. The only case in which this would be useful is when all of the sheep are close together in one single flock.

Sheep with most neighbours

This requires the sheep to be sorted into order of descending number of neighbours. The sheep with the most neighbours is then counted as the centre of the flock, all of the sheep's neighbours are added to the flock. This continues onto the next available sheep with the most neighbours until no sheep are left.

Scanning of points across the field, using points which have most neighbours.

This is more of a brute force method, a predefined density of points would be decided, and then each point across the field is scanned to see how many sheep are within range of it. The points with the most sheep in range are expanded to become the flocks. The efficiency of this would vary greatly depending on the density of the points.

K-Means Algorithm

This is similar to the previous technique, the points are assigned in potentially random starting locations, then sheep are assigned to their nearest point. The points are then moved iteratively so that they are in the average position of the sheep assigned to them, minimising the distance between each sheep and the centre [Hartigan, 1975]. This works well if there is a limit on the number of flocks we want to identify or if a flock is limited by a maximum size.

Hill Climbing

Again, this is similar to the previous two techniques. A random selection of flocks is made, then each one is changed by a small amount to see whether it improves the coverage of the sheep or not. This continues iteratively until no improvements are being made [Luke et al., 2005].

Sub-divide, quad tree, divide and conquer

This idea involves splitting the field up into 4 quadrants, and then doing this recursively. On each step we see if splitting a quadrant up into 4 yields a more accurate placing of the 'box' around the flock.

The consideration of these techniques allowed the following method to be devised:

Flocks as connected graphs where edges are less than a particular distance

This method can be most simply represented by several algorithms:

```
AssignLooseSheep (UsedSheep):
List AllSheep = Every sheep, sorted based on how many neighbours they have,
descending;
For Each sheep not in UsedSheep
                Create Flock flock;
                Call addSheepAndNeighbours( flock, sheep, usedSheep);
                Add Flock f to field;
end for
```

The 'AssignLooseSheep' function takes a collection of sheep that have already been assigned to flocks (used sheep) and then creates a new flock based on each sheep not already assigned. Each time a new flock is created, the 'used sheep' list is updated, until all loose sheep are assigned to a flock.

```
addSheepAndNeighbours ( flock, sheep, usedSheep )
Add sheep to flock;
Add sheep to usedSheep;
Find neighbours of sheep;
For Each neighbour:
        if neighbour is not in usedSheep
        addSheepAndNeighbours( flock, neighbour, usedSheep )
end for
```

The 'addSheepAndNeighbours' function is the recursive step. A sheep and all of its neighbours which are not already assigned, are added to the flock. The function is then called recursively for each neighbour, branching out to encompass all nearby sheep within the flock.

What is not considered here is the need specified earlier for flocks to persist from one step of the simulation to the next. Before assigning loose sheep to new flocks, we must consider sheep which were already in flocks on the previous step, and check whether the flocks have had members join, leave or have merged.

```
maintainCurrentFlocks( usedSheep, flocks )
Create a list sortedFlocks, of flocks descending by size of flock;
For each flock in sortedFlocks:
    Remove any sheep which are now in other flocks;
    Find the main sheep, the sheep nearest to the centre of the flock;
    if main sheep is null
        Remove the flock
    else
        Remove all sheep from the flock;
        addSheepAndNeighbours( flock, main sheep, usedSheep);
        Update the location of the flock;
    end if
end for
```

This algorithm takes all current flocks and reassesses their membership, based on the sheep nearest to the centre of the flock. A key idea here is sorting the existing flocks in descending order based on their size (number of members). This means that bigger flocks are considered before smaller flocks, causing smaller flocks to be merged into bigger flocks when they are close enough.

In order to use these two ideas together, the list of 'used sheep' must be kept when calling both, we use the algorithms as follows:

```
func updateFieldOfSheep()
    List usedSheep = empty list;
    maintainCurrentFlocks(usedSheep);
    assignLoseSheep(usedSheep);
end func
```

A brief description of the implementation details of this algorithm is included in Chapter 7.

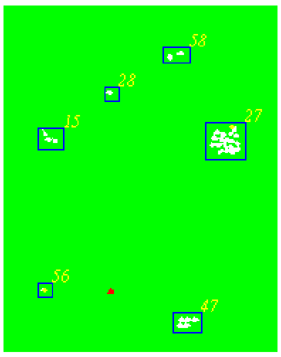


Figure 28: Debug view of the field of sheep.

The above figure (28) gives a 'debug view' of the field of sheep mechanism. The blue boxes represent how the dogs 'see' the flocks, and the numbers represent the unique ID assigned to each flock, allowing the dog to maintain an objective on a particular flock, across multiple steps of the simulation.

6.5 Sheepdog Iteration 3: Herding Sheepdog

The next step was to design a sheepdog that made use of the flock identification techniques described above. The gathering sheepdog that was mentioned above focussed on a single sheep, and made all of its decisions based on only a single sheep and a single target point. Therefore a sheepdog was designed to work with groups of sheep. This sheepdog performs what is known as herding [Lien et al., 2004], which is concerned with moving a flock of sheep from one location to another. This task also encompasses ideas from the gathering behaviours, since flocks may need to be merged, so that they can be herded more efficiently as a single flock. Similarly, if sheep break off from the flock, the sheepdog must gather them again before it can continue herding the flock.

An interesting challenge when looking at driving flocks is how to steer them, many techniques exist, some of which are implemented and evaluated in this section. There is also the need to steer different size and shape flocks, from different distances, at different speeds, to ensure minimum breakups, while still maintaining reasonable progress.

An additional consideration, which is perhaps more valid than for the gathering dog, is the cooperation between multiple sheepdogs when herding flocks. The scope is wider since the task of the sheepdog involves working with multiple sheep, presenting the possibility of multiple sheepdogs formations for herding flocks. Concepts such as obstacle avoidance and steering points discussed for the gathering sheepdog are also used in the herding sheepdog model.

6.5.1 Behaviour Flow Diagram

The following diagram (fig. 29) illustrates the high level decision making process for the herding sheepdog. The process assumes a goal location for the flock. The diagram is split into two clear sections: *herding* the flock, and *merging* the flock, which is the general term we use for gathering sheep and flocks together into one single flock.

The key stages which are investigated in more detail are as follows:

- All sheep in one flock? - This relies on some overall concept of a 'flock' and a method of testing whether or not all sheep in a field are in this single flock. The flock identification techniques described in section 6.4 are sufficient for this.
- Drive flock towards X / Herd driven flock to target flock – These involve deciding how to drive an arbitrarily sized flock [section 6.5.3].
- Choose which flocks to merge – This is concerned with deciding on an efficient method of choosing which flocks to merge, in order to get to a single flock most quickly [section 6.5.4].
- Does driven/target flock still exist? - These are the questions the sheepdog will consider when deciding whether or not to deliberate [section 6.5.2].

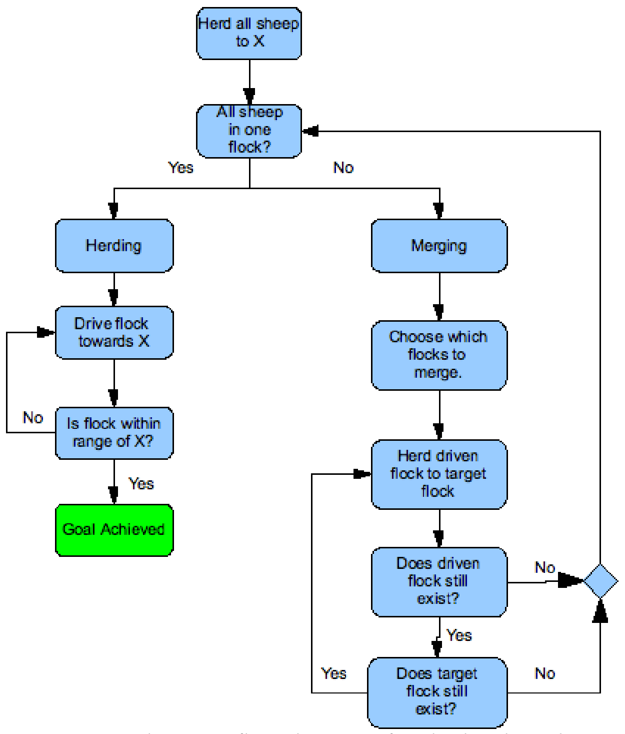

Herd all sheep to X
All sheep in one flock?
Yes
No
Herding
Merging
Drive flock towards X
Is flock within range of X?
No
Yes
Goal Achieved
Choose which flocks to merge.
Herd driven flock to target flock
Does driven flock still exist?
No
Yes
Does target flock still exist?
Yes
No

Figure 29: Behaviour flow diagram for the herding dog.

The following sections elaborate on the ideas presented in the above diagram (fig. 29).

6.5.2 Working with flocks

In the gathering sheepdog model, the decisions were made based on trigonometry on points in a 2D environment. When working with flocks, it was necessary to consider the flock as a bounding box. To use the idea of steering points with a bounding box, a method was designed to work out where the centre of the flock was and where the steering point should be.

There were three possible techniques to use when deciding where the centre of the flock is:

1) The centre of the flock rectangle.

2) The average position of all sheep within the flock.

3) The sheep with the most neighbours in the flock.

The following design assumes the use of the first method, allowing the steering behaviours to work on the centre of the bounding box as its reference point. Since we are working with a rectangle, the distance between the edges of the bounding box and the centre of the flock will vary depending on the angle between the sheepdog and the centre of the flock.

Using the flock identification system

The flock identification system was implemented in such a way that the sheepdogs would obtain a reference to a flock, allowing them to work out where that flock is at any given time. When flocks become merged, one of them disappears. To make this work, the sheepdog checks on each step that the flocks that it was targeting have not become empty, if they have become empty it deliberates again. The pseudo-code for this idea is as follows:

```
func boolean shouldDeliberate()
        if drivenFlock.size() == 0 OR targetFlock.size() == 0
                return true;
        else
                return false;
        endif
end func
```

The flock identification system also provided functionality such as finding the largest flock and returning a list of flocks near to a particular location, to allow the sheepdogs to obtain a simulated *local* view of the surroundings.

6.5.3 Pushing the flock

Once the centre of the flock has been calculated, calculating the the angle required to push the flock at can be reduced to the same problem as pushing a single sheep, as described earlier. The steering distance will be decided by working out the diagonal between the centre of the flock and one of the corners. In the case of a rectangle with a large difference between height and width, this will mean that the steering point can be at different distances from the actual sheep in the flock, depending on the angle of approach.

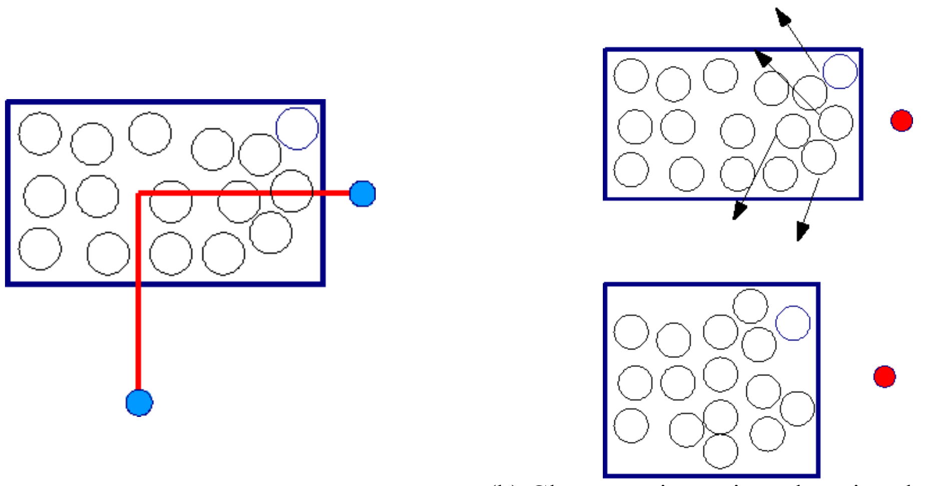

(a) Steering points for a bounding box (b) Close steering points changing the contour of the flock

Figure 30: Pushing the flock

As shown in figure 30(a), although the two steering points are identical distances from the centre of the bounding box, they are at different distances from the actual sheep. This approach was chosen since it encourages the flock to be compacted as it is steered. If the flock is long and thin, and is steered from the thin end, the steering point will be close to the sheep, forcing the contour of the flock [Lien et al., 2004] to become more regular (fig. 30(b)).

Later in this section we consider a mechanism for varying the steering point based on how receptive the flock is to the sheepdog's advances, using the concept of emotion in the sheepdog agent.

Straight steering

Due to the way the flock identification was implemented, it was possible to use the same algorithm for straight steering a flock as it was a single sheep, the sheepdog simply worked on the centre point of the flock, with a slightly larger steering distance, dependant on the size of the flock.

Side-to-side steering

A more sophisticated steering behaviour, as described by Lien et al. [2004], was implemented. The dog alternates to the left and right of the steering point, in order to prevent the flock from breaking up.

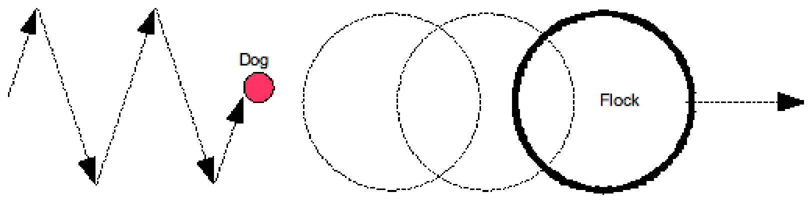


Figure 31: Side-to-side steering

The width of the side-to-side excursions is dependent on the size of the bounding box for the flock being driven. The steering points in this case were on each side of the flock. Since the sheepdog was not pushing directly at the flock, the steering points were placed closer to the flock to ensure progress is made.

The algorithm for this type of steering is as follows:

```
State:
lastSteeringDirection = left;
func Step() // on each step of the simulation
    Calculate the steering point.
    Draw a line, normal to the intended motion of the flock, passing through the
    steering point.
    if lastSteeringDirection == left
        Move towards leftmost part of the line.
        if reach the end of the line, lastSteeringDirection = right;
    else if lastSteeringDirection == right
        Move towards the rightmost part of the line.
        If reach the end of the line, lastSteeringDirection = left;
    endif
end func
```

The length of the line is varied, proportional to the size of the flock being driven, to ensure that the sheepdog reaches the extent of the flock on each alternation.

6.5.4 Merging Flocks

The above discussion of how to drive a flock assumes that we are working with one single flock which can be driven as a single entity. We must consider the case whereby the flock becomes separated during herding or even when it was never a flock in the first place. The concept of merging flocks is similar to that of the gathering sheepdog described above, except that flocks are gathered rather than single sheep. Ideas from the gathering dog can be incorporated to make a more sophisticated sheepdog which can both gather and herd sheep towards a goal using the flock identification techniques we have discussed in section 6.4.

There were many different algorithms which could be considered for merging an arbitrary number of flocks, the study of this type of problem could be a project within itself. Several different algorithms were implemented for this project, the code for these algorithms can be found on the attached CD:

1) 'Biggest Flock' - Drive all other flocks to the biggest flock, similar to that of the gathering dog.
2) 'Travelling Sheepdog' - Herd the flock closest to the sheepdog to its closest flock, repeat with newly merged flock. Ideally, this would actually involve making n-1 journeys between n flocks, travelling the shortest distance possible, this is very similar to the famous Travelling Salesman Problem [TSP, 2008]. For this project, a greedy algorithm is used which simply takes the next shortest possible route each time.

3) 'Nearest Two' - Find the two flocks nearest to the sheepdog, herd the smaller of these two to the larger.

4) 'Loose Sheep First' - Send all flocks smaller than the flock threshold of 5 to a nearby flock, then use method 3.

Alternatively when working towards a goal, the gathering task used a different approach:

1. Merge flocks to the flock which is nearest to the goal and then herd the flock to the goal.

A key point in the effectiveness of these algorithms was related to the way in which the herding dog deliberates on its goals. As shown in figure 29, each time either the driven or target flock becomes merged with another flock, the sheepdog will deliberate. This is a form of single-minded commitment [Wooldridge, 2002], since the sheepdog will continue to drive one flock to another while it still believes it is possible to combine these two flocks and while it has not yet achieved the goal. A slight modification to this behaviour was to put an upper limit on the number of steps for which the sheepdog will attempt to achieve a particular goal. This is to prevent the sheepdog getting in to any impossible tasks due to deadlocks or other unforeseen emergent behaviour in the simulation.

6.5.5 Multiple Herding Dogs

When considering multiple herding dogs, we have to consider how to work with flocks as a single entity. This calls for some more sophisticated cooperation, since the task of herding a flock can potentially benefit from having two or more dogs working on the same flock.

Sheepdogs consider how many other dogs are nearby when they are trying to herd a flock. If there are other sheepdogs within a certain range, the sheepdogs would expect them to help. Therefore, each sheepdog near to the flock will have an idea of how many others there are, and therefore where they might arrange themselves in order to drive the flock.

6.5.6 Multiple Sheepdog Formation

As Barry and Dalrymple-Smith [2005] showed, using the basic flocking rules to attempt to create a multiple predator formation was an attractive and elegant solution. However it was decided that the obstacle avoidance issues as described in section 6.3.5 would make it difficult to work this way. Therefore, some explicit formation knowledge had to be designed so that each sheepdog could make a decision on where to go based on local information.

Line formation

To be able to form a line to drive the flock, without any external coordination, the following algorithm was used for the sheepdogs:

```
CountDogs =  number of dogs in area.
If no other dogs
    steer as normal
else
    Calculate steering point.
    Draw a line at the normal to the line which intersects the steering point and the
centre of the flock.
    Divide the line into (CountDogs + 1) segments.
    Each point between segments is a steering point.
    For each steering point
        If no dogs near to steering point, move towards steering point.
    end for
endif
```

During the approach to the steering points, the dogs also exert a separation steering behaviour on each other, to try and stop them from converging towards the same steering point.

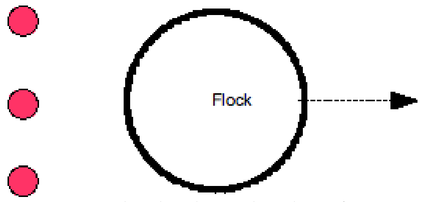


Figure 32: Multiple sheepdog line formation

The above diagrams shows three dogs in a line formation, aiming to drive the flock to the right.

6.5.7 Herding Sheepdog Evaluation

The herding sheepdog worked effectively on flocks of sheep as well as single sheep. Since the sheepdog worked with arbitrary sized flocks, the problem of gathering sheep could also be accomplished by the herding sheepdog, using the merging algorithms described in section 6.5.3. This, along with the behaviour shown in figure 29 allowed the sheepdogs to herd flocks to form a single flock, and also to fix any breaks in the flock whilst driving.

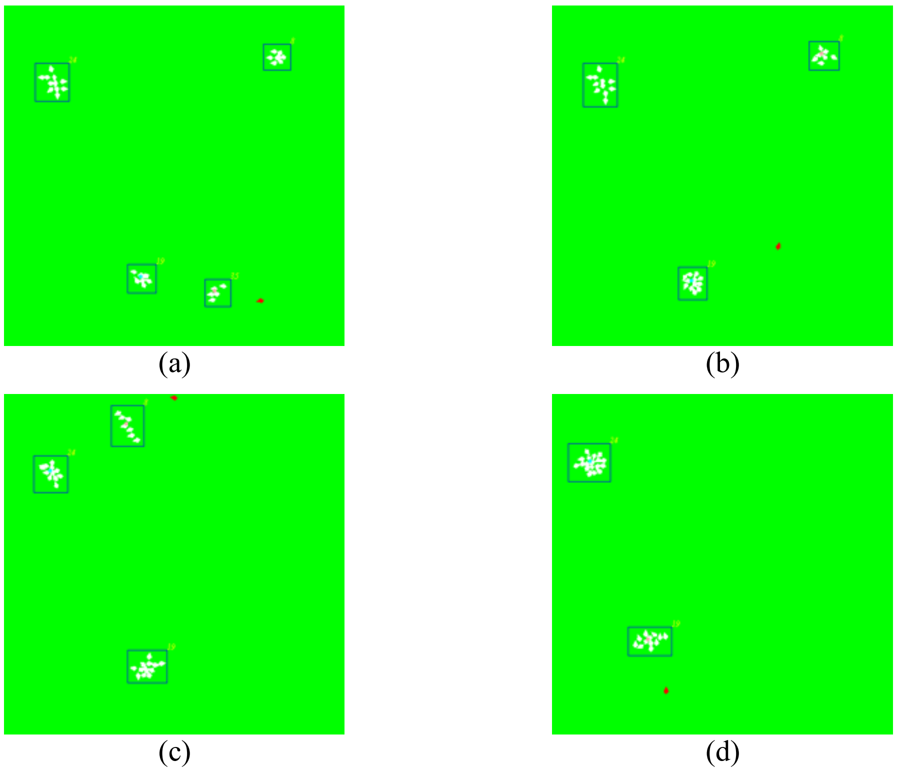


Figure 33: Merging flocks

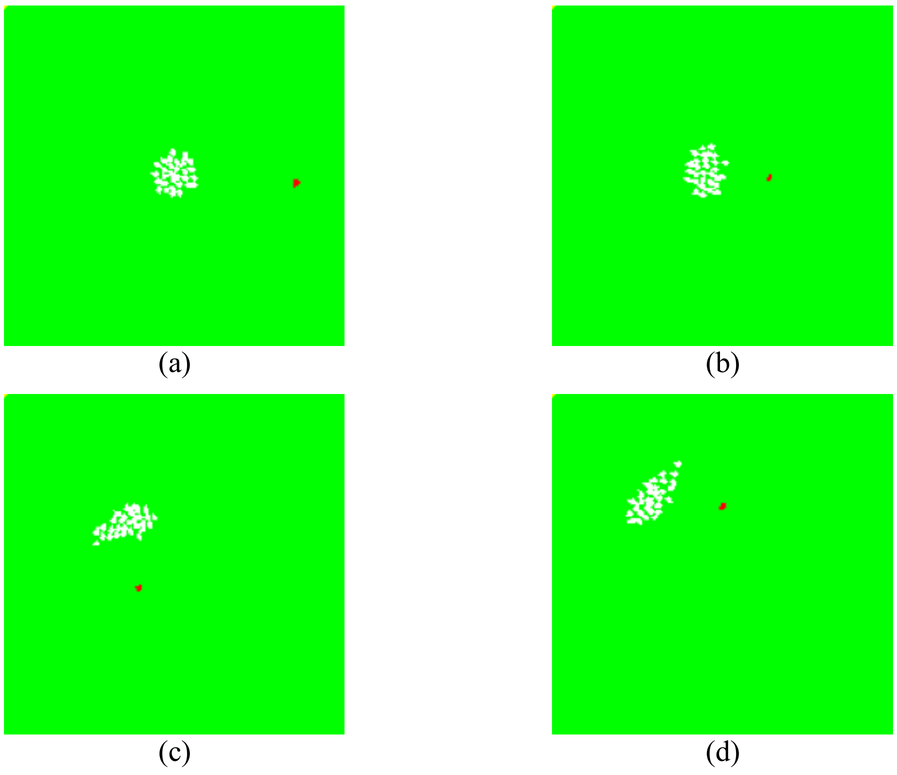


Figure 34: Side-to-side steering

Figure 33 shows, with debug information, a single sheepdog merging various flocks together using straight steering. A side-to-side steering technique for flocks was implemented, it proved to be effective at maintaining flock cohesion but less progressive than the straight steering technique.

In figure 34 we can see the dog (red) performing a side to side motion behind the flock, causing the flock to move towards the top left corner while also staying as a single flock.

Arc steering would have been the next step in terms of steering techniques. While experimenting with side-to-side steering, an arc steering style behaviour emerged by setting the side-to-side steering points across opposite sides of the flock, and using obstacle avoidance when approaching them.

Simple, spontaneous coordination was developed for multiple sheepdogs when herding the same flock, as well as being used in the same way as the gathering dog, to merge flocks together more quickly.

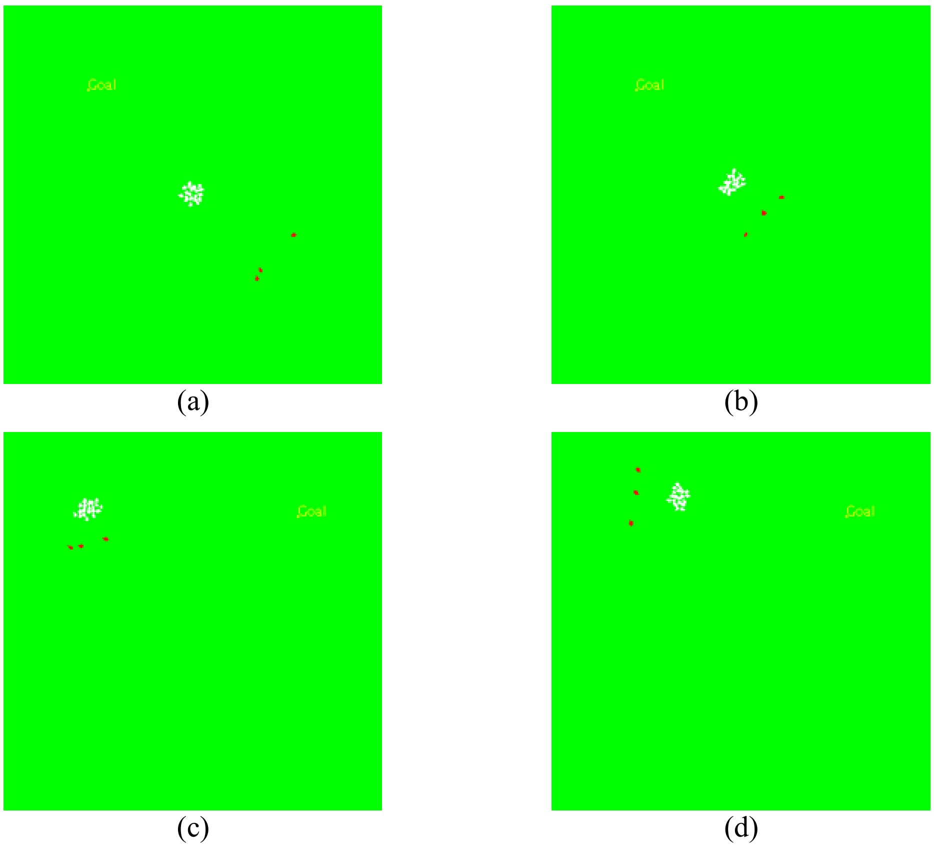


Figure 35: Multiple herding dogs

Figure 35 demonstrates the coordination between three sheepdogs, showing them spreading out to drive the flock in a controlled manner. Side-to-side steering did work for multiple sheepdogs, but was not so useful, since the dogs appeared to follow each other single file. This could have perhaps been improved by making the dogs perform a side-to-side steering behaviour on their own personal steering point within the line formation.

This iteration signalled a reasonable level of functionality for the sheepdog model, providing various techniques for gathering and herding flocks of sheep.

6.6 The Oracle

The 'oracle' was a concept designed to abstract the goal setting and goal checking mechanisms that a shepherd may perform.

6.6.1 Goal Setting

As mentioned above, for multiple sheepdogs, it is necessary for them to be able to work on the same task without an external coordinator. However, when trying to herd a flock to a goal, there has to be some indirect communication, so that the sheepdogs are working towards the same goal. This was implemented as an 'oracle', that is, a global entity in the simulation that generates a list of goals, assesses when goals have been achieved and informs sheepdogs when this happens.

Similarly, when gathering sheep, a specific location can be specified where the sheep must be gathered to. The gathering dogs must also be able to access the oracle, so that they can work towards the same goal.

6.6.2 Goal Checking

In the same way, it was necessary for there to be a central entity responsible for checking when a goal has been achieved. This is mainly useful for testing how long it takes for goals to be achieved, when running tests such as those in Chapter 8.

The oracle was implemented as a thread-safe singleton class, so that multiple dogs accessing it on different threads would see the same goals and would be correctly notified of the successful achievement of goals. Using the observer design pattern [Gamma et al., 1996], the dog agents register as observers of the oracle class, so that they receive new goals when they are decided, and are notified when goals are achieved.

6.7 Sheepdog Iteration 4: Personality and Emotion

As a final extension to the project, personality and emotion was investigated for the herding sheepdog, but not the gathering sheepdog, since it was found that the herding sheepdog could accomplish everything that the gathering sheepdog could.

6.7.1 Sheepdog Personality

A simple personality was used for sheepdogs to vary their behaviour and interaction with sheep. From the background research, it is evident that different breeds of dog work with sheep in subtly different ways. As identified by Downe [2005], sheep have respect for certain sheepdogs, and are more compliant when being worked by dogs that they respect. To model this, a single value was used to represent the amount of respect a sheepdog will gain from sheep.

A sheepdog with a higher respect value will cause a sheep's fear counter [section 5.3.2] to increase more rapidly, and therefore will flee more easily. Conversely, a sheepdog with a lower respect value will spend a lot of time staring at the flock in order to build up enough fear in the sheep for them to flee.

6.7.2 Sheepdog Emotion

Currently, if we have two or more dogs attempting to drive the flock in the same direction, everything works as expected. If however, the dogs are trying to drive the same flock to separate goals, because they have different plans for how to round up the sheep, it is possible for them to get into deadlock. Using ideas covered by Murphy et al. [2002] it is possible to use emotion to break the deadlock.

Emotion was used in the herding sheepdog model as a means of preventing deadlocks when a flock would not move and the sheepdog had reached its steering pivot. The emotion used to break these deadlocks was frustration. This was modelled using the following assumptions:

- If the sheepdog is having to move slower than a certain speed in order to drive the flock, then it will become frustrated.
- The more frustrated the sheepdog is, the more aggressive it will become, by getting closer to the flock.
- When the flock is moving at a satisfactory speed, the sheepdogs frustration will decrease

These ideas have the following effects on the sheepdog behavioural algorithms.

```
SteeringPivot()
        return Max(steeringPivot – frustration, 0);
end func
```

This simply shows how the steering pivot can vary depending on the frustration, but can never become a negative steering pivot, since this would mean trying to enter the flock to steer it.

The following pseudo-code shows how the frustration emotion is updated on each step of the simulation.

```
Step() // on each step
    speedRatio = currentSpeed / maxSpeed;
    if speedRatio < X
        increase frustration;
    else
        decrease frustration;
    endif
end func
```

We use the comparison of the speed ratio with a constant X. X varies depending on the personality of the sheepdog, a more patient sheepdog will have a lower value for X and will be more comfortable moving at a slower speed.

6.7.3 Evaluation of Sheepdog Iteration 4

All of the emotion and personality functionality was implemented at a higher level in the system so that it could be shared by all types of sheepdog. The emotional subsystem was updated on each step of the simulation and kept separate from the rest of the dog behaviours.

Frustration was an interesting addition to the sheepdogs behaviour. It did give the impression of the sheepdog having some level of determination to drive the flock at a particular speed. The idea had to be turned off for side-to-side steering, since the dog is always moving at its maximum speed, alternating on each side of the flock, so in this case the frustration counter would never be incremented.

For multiple sheepdogs it also caused problems. Since we were working on the assumption of spontaneous coordination, each sheepdog had to make an estimate as to where the other sheepdogs were lining up in the formation. However, when using an emotion-affected steering point, each dog had different ideas of where the steering point was and thus could not differentiate effectively between 'free' and 'used' steering points.

It is difficult to illustrate the effectiveness of the emotion and personalities qualitatively, therefore the tests in Chapter 8 will provide a statistical view of how emotion affected the task of herding sheep.

Chapter 7, Implementation Details

This section includes class diagrams and a brief system overview as well as some key implementation details.

7.1 Environment Class Diagram

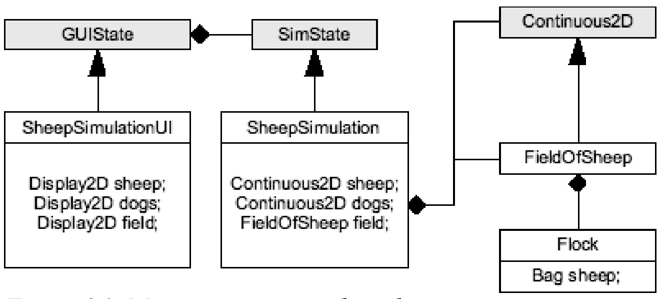


Figure 36: Main environment class diagram

The above section of the class diagram shows how animals were stored in the environment. *SheepSimulation* is the main simulation class which extends the Mason *SimState* class, containing functionality to start and stop simulations as well as the various model parameters. In the *SheepSimulation* class, 'sheep' and 'dogs' were *Continuous2D* fields, which stored untyped objects in *hashmaps*. We store two separate *Continuous2D* environments in the *SheepSimulation* class to differentiate between sheep and sheepdogs, since the *Contiuous2D* environment stores untyped objects. The hierarchy of objects which were stored in these fields is shown below in figure 37. The logic to work out which *Continuous2D* to query each time is handled by the agents themselves.

FieldOfSheep is an extension to the *Continuous2D* class, by taking information from the 'sheep' environment on each step, a series of flock objects are built and maintained. It was implemented as a *Steppable* object, effectively making it an agent. It provided an interface for the Dog classes, allowing them to obtain flock positions, sizes and a list of sheep in each flock. On each step of the simulation, the 'field of sheep' object received the positions of all the sheep in the field and updated the locations of its Flock objects appropriately.

Dogs had to check if a flock that they were working with still had a size of greater than 0. If the flock size was 0, the dog was able to deliberate on its goals, choosing a different flock to herd.

A separate *Display2D* object is used for each of the three main fields described above, allowing them to be layered when displayed in the GUI of the model.

7.2 Agent Class Hierarchy

The following diagram illustrates the hierarchy of agents within the system:

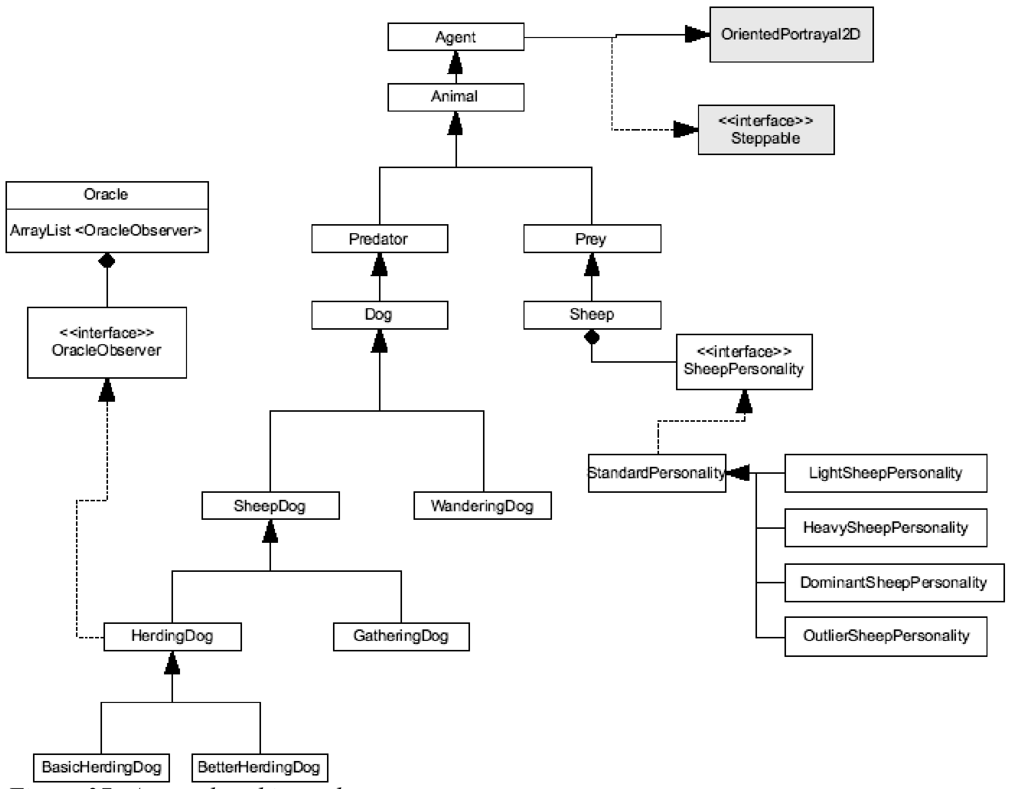


Figure 37: Agent class hierarchy.

The top most class is the Agent class, this extends 'OrientedPortrayal2D' from Mason, which gives it the ability to be displayed as an oriented object, that is, one which points in a given direction. The Agent class also implements the *Steppable* interface, allowing it to be scheduled in the simulation, so that its 'step' method is called on each step of the simulation. As mentioned above, the simulation environment is divided into two *Continuous2D* fields, one intended for storing Predators and the other for Prey.

The Dog class is extended by several others, WanderingDog represents a simple randomly wandering agent as described in section 6.1. SheepDog contains some of the behaviours common to both the HerdingDog [section 6.5] and GatheringDog [section 6.3], such as movement towards a steering point. The HerdingDog class implements the OracleObserver interface, allowing it to register with and receive task information from the oracle, as described in section 6.6.

The Sheep class contains a class implementing the SheepPersonality interface. The SheepPersonality interface requires methods that provide information on how a particular sheep will flock and react to predators. The five personalities of sheep [section 5.3.4] are represented by a hierarchy implementing the SheepPersonality interface.

7.3 MASON Features

The project was implemented using the MASON multi-agent simulation toolkit [Mason, 2008]. Appendix A outlines some of the general MASON features that were used, the following implementation details were particularly relevant to the project.

7.3.1 Random Numbers

MASON includes a third party random number generator using the *Mersenne Twister* algorithm [Matsumoto and Nishimura, 1998]. The key reason for using this as opposed to the standard Java random number generator is that it is fully synchronized and is suitable for use in a multithreaded environment. The standard Java random number generator is seeded solely on system time and therefore is not suitable for use in a multi-agent system where many agents may be requesting random numbers at similar times.

The implementation included is the MersenneTwisterFast class, it provides an identical interface to that of the standard Java random number generator as well as some extra functionality. It allows probabilities to be specified when requesting a random boolean, this was useful when an action is required to be performed every x steps on average.

7.3.2 Animal Perception

MASON provides a useful feature for looking up objects within Continuous2D environments. Given a particular point, it is possible to look-up all objects within a certain radius of that point.

```
getObjectsWithinLocation( location, range )
```

The important condition when using this function is that it returns *at least* those objects within the specified distance. Therefore, it is possible for this function to return objects that are located outside of the specified distance. An extra check was added to the code when calling this function, in order to guarantee that all objects returned were strictly inside the range. This function was used to simulate the vision of animals within the system, aligning closely with the idea of 'neighbourhood' proposed by Reynolds [1987].

7.4 Summary

Full source code can be found on the attached project CD. A brief look at the technologies used can be found in Appendix A.

Chapter 8, Testing

8.1 Introduction

Due to the nature of this project, the most effective way of assessing how well the model has addressed the original aims was by either seeing the model in action or by seeing a video of the model. This was not possible within this report, therefore a collection of screen shots throughout the development and an exhaustive array of results and graphs from the model help to paint a picture of how the model really worked. This section focuses on performing a series of parameter sweeps on the model, with the aim of finding some interesting data to prove that the model has met the original objectives.

8.2 Tuning the Model

As mentioned throughout the iterative development cycle, a lot of tweaking and experimenting with constants was performed to try and get the desired behaviour. Most of these constants and values were extracted into the model interface itself as simulation parameters, allowing the user to modify them at runtime. See Appendix B for a guide to the various parameters and how they affect the model.

8.3 Test Cases

The main tests that were run involved looking at how different types of dog worked with different types, personalities and numbers of sheep. There are few interesting tests that can be run that focus solely on the sheep behaviour without a sheepdog.

8.3.1 Parameter Sweeps

Parameter sweeps involve running the simulation many times, varying one parameter each time. From one sweep, a set of results are obtained, allowing us to choose a starting point when varying another parameter, and so on. Throughout this testing section we perform parameter sweeps, allowing us to focus in on the more interesting parts of the model as the testing progresses. Therefore, the amount of data collected for the later tests is smaller, since we have a better idea of what we are looking for and how to find it.

8.3.2 Initial Setup

A variety of different methods of setting up the tests were used. These methods had to consider whether or not the initial state of the model was repeatable, so that it could be run with slightly different parameters.

Sheep Setup

Single Herd – All sheep placed in the environment at a single point, the separation forces of the sheep will cause the flock to expand to its natural size, used for testing herding behaviour.

Random – Each sheep is placed randomly, used to test the effectiveness of a sheepdog over many different initial states. Due to the way MASON generates random numbers, it is possible to specify a random number seed which generates the same sequence of random numbers each time. This allows the same initial setup to be used with various types of sheep.

Dog Setup

The dog can either be placed:

- Randomly – At some point in the environment.
- In a particular place e.g. top left, top right, bottom left, bottom right, centre.

8.3.3 Completion

The environment was able to check for goal completion on each step of the simulation, stopping the simulation and writing the results to file when completed. The goal conditions were either:

One Flock – Gather the sheep into a single flock.

Series of Goals – Herd the sheep through a series of goals.

8.4 Model Testing

As well as the testing throughout the iterative development and the general observations made above, more statistical approaches were applied to the model to investigate how changing certain parameters affected the results.

The experiments were run a number of times, using MASONs command line arguments to specify the number of repetitions and the random number seed to use. This meant that there was no GUI while running these experiments, which increased the speed dramatically. MASON does not provide a simple way of changing parameters between each sweep without writing batch files to run the simulation from the command line with many command line parameters. Instead the model code was altered before each experiment, to manually set up the model parameters.

8.4.1 Simple Gathering

The first model test looked at the most simple form of meaningful interaction between the sheep and sheepdogs. The field was initialised with a random positioning of sheep, the goal of the sheepdog was to gather the sheep into a single flock, where this flock was did not matter. The number of sheep was varied between 5 and 40, in increments of 5. The sheepdogs used the simple gathering algorithm [section 6.3]. One sheepdog used obstacle avoidance of sheep when approaching whilst the other did not.

The results were obtained by running 50 simulations for each number of sheep, for each sheepdog. The same random number seeds were used in the simulations for both dogs.

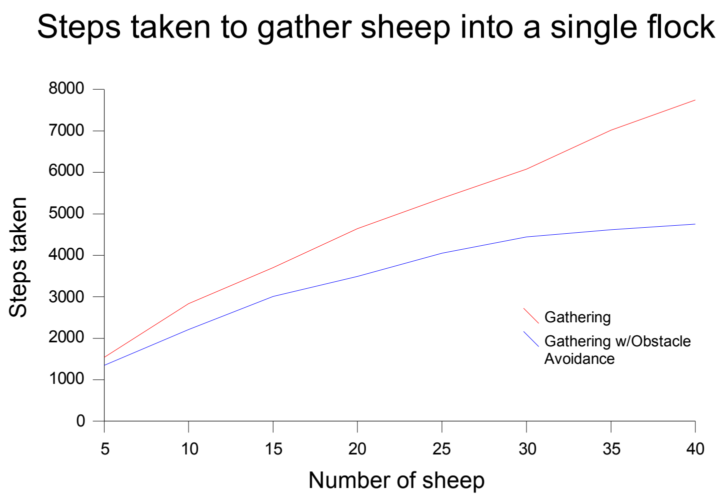


Figure 38: Gathering with and without obstacle avoidance.

The graph clearly shows that using obstacle avoidance produced a more efficient sheepdog. This difference becomes more pronounced as the number of sheep increases. This was observed to be due to the amount of times the sheepdog *without* obstacle avoidance breaks up already formed flocks as it approaches other sheep, the probability of this happening increases as the number of sheep increases.

The graph begins to flatten out for the sheepdog with obstacle avoidance as the number of sheep increases. This is perhaps due to the fact that, gathering of multiple sheep is occurring as emergent behaviour, even though the sheepdog is only acting on a single sheep at a time. As the field becomes fuller there is a greater chance of unknowingly picking up more sheep along the way, when only intending to gather a single sheep.

8.4.2 Multiple Gathering Dogs

From the previous test, it was clear that the gathering sheepdog with obstacle avoidance was more efficient, therefore subsequent tests will only use sheepdogs with obstacle avoidance on approach.

This experiment investigates whether or not the simple cooperation rules added to the gathering sheepdog can help increase the speed in which the sheep are gathered into a single flock.

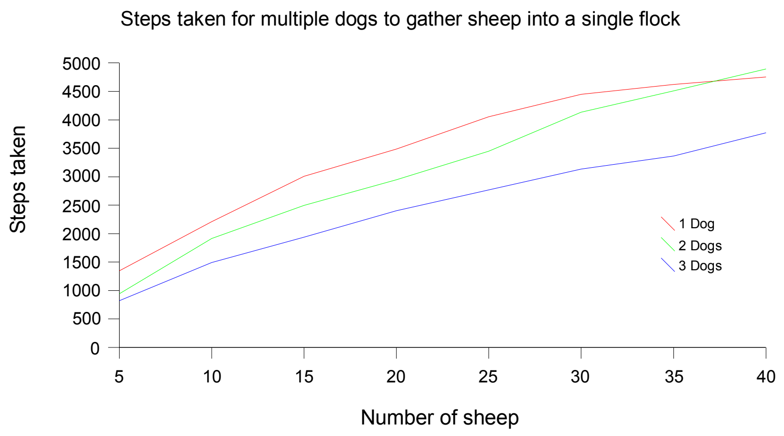


Figure 39: Multiple gathering sheepdogs

The graph (figure 39) shows that when the number of sheepdogs is increased, a reasonable reduction in the number of steps required to gather the sheep occurs. However, there is an interesting point at 40 sheep where 1 dog is actually more efficient than 2. It was hypothesised that the algorithm for deciding which sheep to herd was not totally effective when going from one to two dogs, due to the possibility of getting into 'deadlock' situations where the dogs are both trying to do the same thing in opposite direction. When increased to three, this clearer improvement can be explained since having three dogs breaks the 'straight line' style deadlocks, as described in section 6.3.5, and for smaller numbers of sheep, gave the same order of speedup that was observed between 1 dog and 2 dogs.

8.4.3 Multiple Gathering Dogs with Improved Cooperation Algorithm

In order to improve the cooperation between multiple gathering dogs, layering was added to the sheepdog when in an idle state, similar to the ideas of Brooks [1986]. This removed the potential for deadlocks caused by competition between random movement and avoiding other dogs. The algorithm for deciding which sheep to herd was made slightly more complex, working on each sheepdog's local environment rather than the environment as a whole. This enabled cooperation to occur effectively by increasing the probability of multiple dogs working on separate sheep, rather than wasting time both focusing on the same sheep.

When we compare the efficiency of a single dog with the two different algorithms for deciding which sheep to herd, we see that the method designed for cooperation actually slows down the single dog.

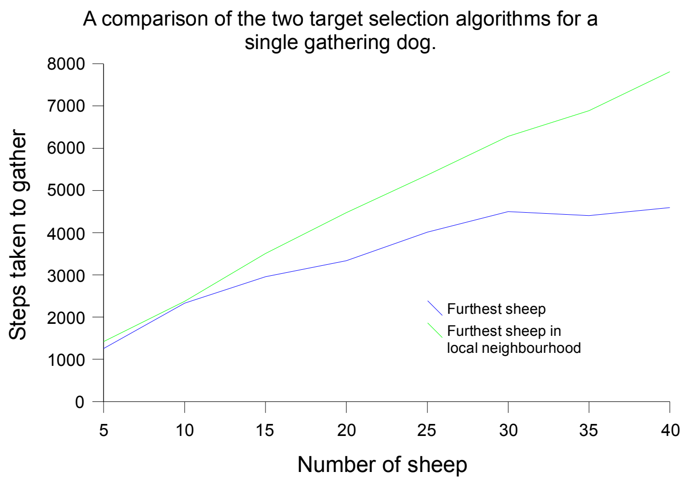


Figure 40: Comparison of gathering algorithms

This shows that the 'furthest sheep' (blue line), algorithm should be used when only a single gathering dog is being used. In future tests, a conditional was added to the gathering dog, which made it choose the gathering algorithm based on whether or not there were other sheepdogs nearby. Using the improved cooperation algorithm, we compared the efficiency of gathering with between 1 and 8 dogs:

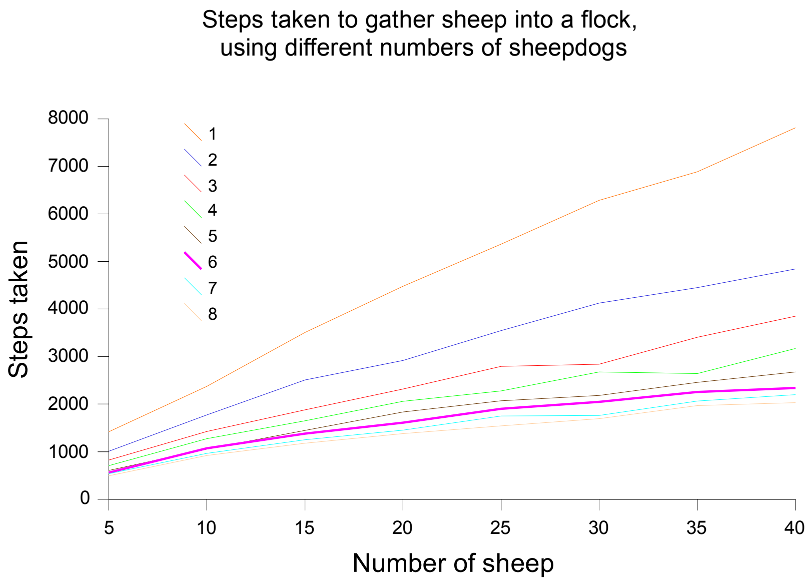


Figure 41: Gathering sheep with multiple dogs and improved cooperation

This graph (fig. 41) clearly shows speedup when more sheepdogs work together. As the number of sheepdogs increases, the amount of improvement is reduced. For this particular size of environment, with between 1 and 40 sheep, anything above 4 sheepdogs has a relatively small gain.

8.4.4 Gathering sheep with different personalities

So far we have focussed on gathering sheep with 'standard' personality traits. The next test looks at the effectiveness of gathering the sheep with various personalities.

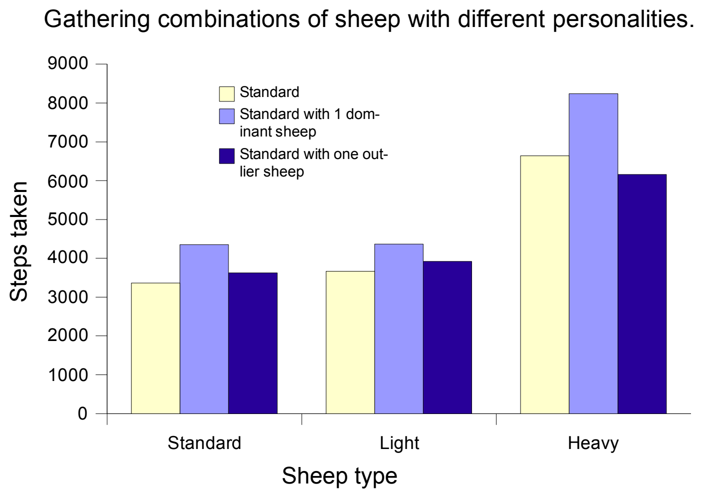


Figure 42: Gathering sheep with different personalities

The x-axis in the above graph (fig. 42) represents the type of sheep that the flock was made up of. The bars in each section represent the whole flock being of that type (cream), the addition of a dominant sheep (light blue) and the addition of an outlier sheep (dark blue).

We can see that for the standard flock, the addition of a leader sheep makes the task harder for the gathering dog, because the dominant sheep can lead the flock away from its intended location and will stand up to the sheepdog. Similarly, the addition of an outlier makes it marginally more difficult, presumably because there is an additional sheep that is more difficult to gather and more likely to wander away from the flock once gathered.

The same pattern can be seen for the light sheep, albeit slightly more difficult in each case, because of the increased flee range. Since the sheepdog is using the obstacle avoidance behaviour as described above, it is unlikely to disturb the flock, meaning that the small difference in gathering light sheep is perfectly reasonable.

The heavy sheep is shown to be much more difficult to gather than the others, especially when in the presence of a dominant sheep. This reflects the stubbornness and reduced flee response in a heavy sheep. The fact that the heavy sheep has the biggest difference when combined with a dominant sheep highlights the desire to be with a dominant sheep. An interesting result is that the addition of an outlier sheep actually makes the gathering task easier. This was found to be because of the high desire to flock in heavy sheep causing them to stay close, even to an outlier sheep, and since the outlier sheep is gathered similarly to a standard sheep, gathering this sheep can help gather the flock as a whole.

8.4.5 Gathering the Flock with Flock Identification

The third iteration [section 6.5] of the sheepdog introduced an agent that could work with flocks [section 6.4] as bounding boxes rather than single sheep. The primary aim of this was to allow the dog to herd entire flocks to specific locations, however it was hypothesised that it should be faster at gathering sheep than the 'gathering dog', because of its ability to work with flocks.

The following diagram illustrates the steps taken to gather a field of sheep into a single flock. Each line represents a different gathering method, with the latter three being variations of a herding sheepdog.

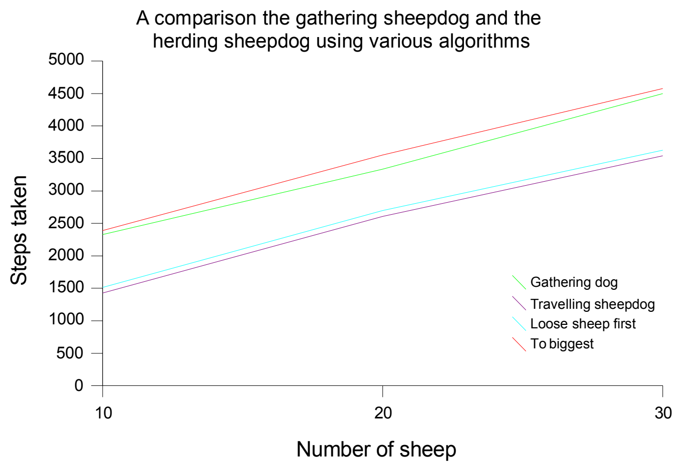


Figure 43: Gathering using flock identification

This shows a very clear difference between the gathering dog and the herding dog, which was used with three separate gathering algorithms. The 'To Biggest' method was slightly slower than the gathering dog, which was unexpected. Possible explanations for this were that the herding dog was deliberating too often or choosing target flocks which were a large distance a way each time, causing a lot of unnecessary travelling. The 'travelling sheepdog' and 'loose sheep first' methods represented a significant (20-30%) improvement to the gathering effectiveness.

It was observed that the 'travelling sheepdog' method, although being the fastest was not necessarily the most logical or reliable. The general idea is that the sheepdog will visit each flock once, collecting them as it goes. Intuitively this means that the size of flock being driven will become larger as time goes on, making it more liable to break up. Therefore, the 'Loose Sheep First' method, which gathers all sheep into flocks of size at least 5, before performing 'Nearest Two' flock merging as descried in section 6.5.3.

8.4.6 Herding the flock

The following tests focus on using the herding dog for its original intention, which was to work on flocks as a whole in order to move a single flock to a specified location. In order to test this, sheep are inserted into the simulation at a single point, so that they start off in a flock. The sheepdogs are then given a series of goals, which the sheep must pass through and arrive at.

It is important to consider the speed in which this task can be performed, as well as the number of times the flock may break up, since it is desirable to both herd the flock quickly and also to reduce the amount of stress the sheep are under [Sheep, Animal Corner, 2008].

The following diagram (fig. 44) shows the course that we used in each of the tests, the sheepdog must herd the flock to each of the goals in turn.

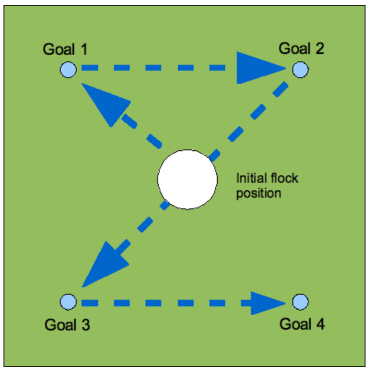


Figure 44: Herding goals

The following test was run 20 times for steering distances of 15, 20 and 25 units. The steering distance defines how far away from the flock the sheepdog will stand when trying to push the flock.

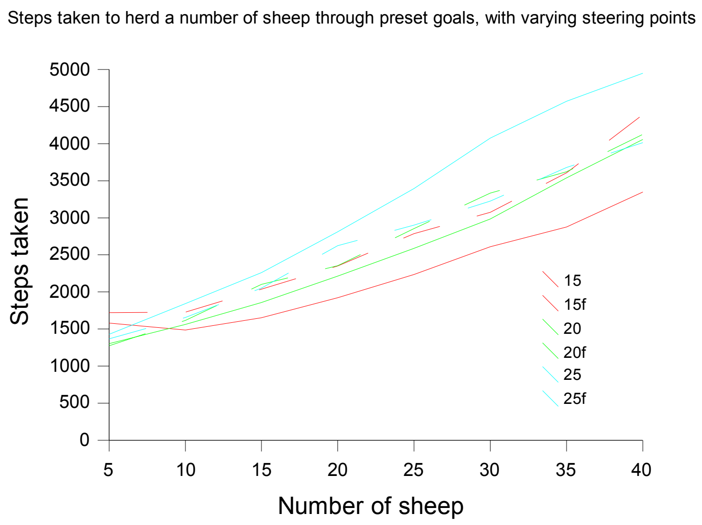


Figure 45: Herding different sized flocks, with and without frustration

The above graph (fig. 45) shows how the different steering distances affect the time taken to herd a flock through the goals shown in figure 44. The immediate conclusion from this graph is that 15 is best steering distance, since it outperforms the other steering points for all but the smallest number of sheep. Smaller steering distances were experimented with, but it was found that they caused the flock to break up far too often, making it almost impossible to herd the sheep.

An interesting point in this graph is the use of emotion for the sheepdog. The dashed lines, labelled with 'f', show the results when the sheepdog was using frustration as a means of varying the steering point. This method did not produce any more efficient herding, however it did show that using this emotion, the steering distance becomes less important, since all 3 emotion based results are very close. This idea was examined further with an even larger flock, the following graph illustrates the effect of varying the steering point, with and without emotion, on a large flock of 50 sheep.

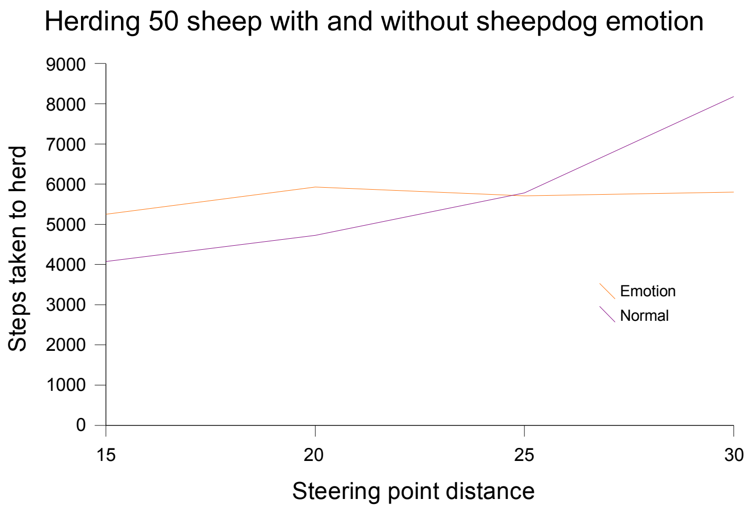


Figure 46: Herding fifty sheep with and without emotion.

This again outlines the fact that a closer steering point is more effective, but when using emotion it makes little difference where the steering point is chosen.

Herding so far has been performed using standard 'straight steering', where the sheepdog pushes the flock based on its perceived centre point towards the goal. The following tests investigate the effectiveness of side-to-side steering as well as multiple dogs working together to steer a flock.

8.4.7 Steering Techniques

The following tests were performed 20 times, on between 10 and 30 sheep. Straight steering as seen previously was tested against side-to-side steering as well as multiple sheepdog formation steering, using between 2 and 4 dogs.

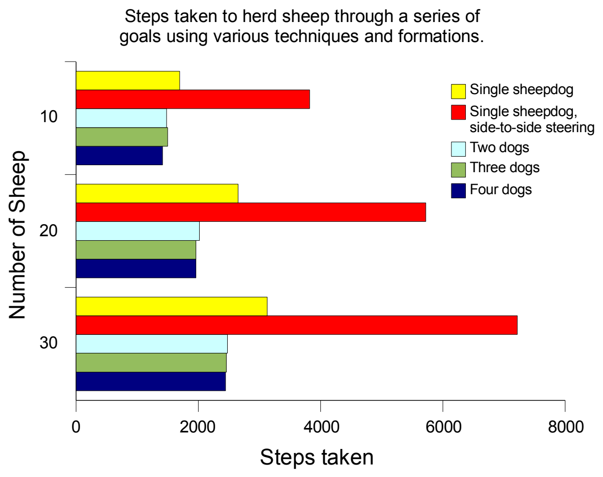


Figure 47: Comparison of steering techniques

The above graph (fig. 47) shows that a speedup is gained in each case, by increasing the number of sheepdogs, with a reasonable difference between 1 and 2 sheepdogs. However, above 2 sheepdogs, the difference is marginal. Most noticeable from the graph is that the side-to-side steering technique is significantly slower than straight steering, since the sheepdog has to cover more ground in order to herd the flock. To understand these results further, another test was performed to count the number of times the flock broke up into one or more flocks during the herding, this gives us an idea of how well the sheepdog(s) are controlling the flock.

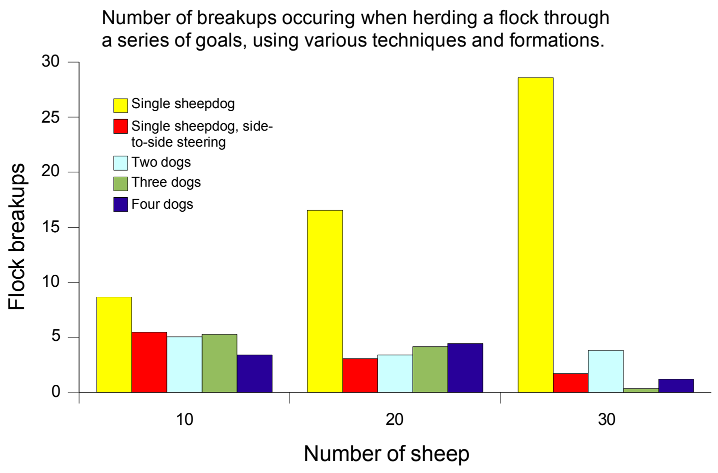


Figure 48: Flock break ups when using different steering techniques

We can see from figure 48 that a single sheepdog performing straight steering causes more flock break ups than any other technique, this is more evident as the size of flock increases. The graph suggests that the use of side-to-side steering can control the flock just as well as using multiple dogs, albeit at a much slower speed (fig. 47). There is evidence that two dogs perform much better than one, however, for more than two dogs there is little correlation between the number of dogs and how 'controlled' the flock is. There appeared to be a minimum amount of interference with the flock which could not be avoided. when the goal changed the sheepdogs all tried to get round the flock to the new steering points, potentially breaking up the flock.

8.4.8 Herding different sheep personalities

As with the gathering task, tests were performed to see how well the sheepdog could herd flocks made up of different personalities of sheep. Flocks were made up of 20 sheep. Flocks of standard, light and heavy sheep were tested as well as mixtures of 19 standard sheep with 1 leader and 17 standard with 3 outlier sheep. The results obtained were less easy to explain than previous tests.

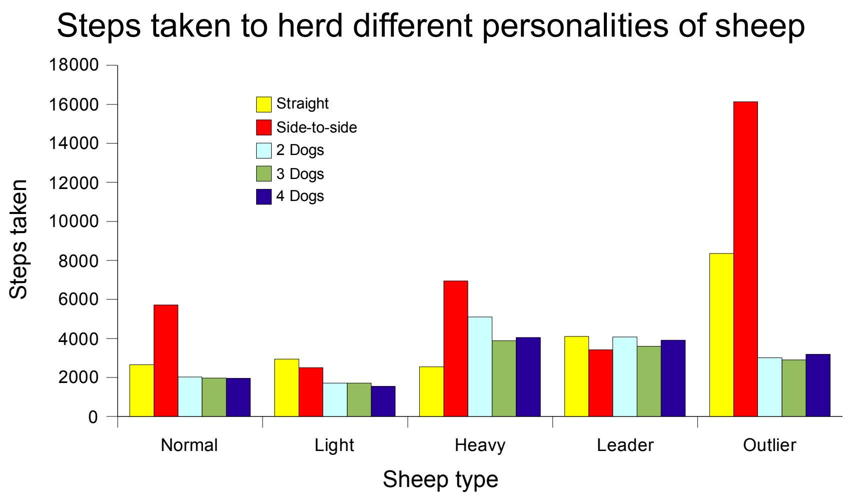


Figure 49: Herding different personalities of sheep

Here we can see that for the light sheep, all methods apart from the *single dog with straight steering*, worked faster than with the standard sheep. This is because the more controlled method of using side-to-side steering or multiple sheepdogs is able to keep the flock together while making use of the light sheep's higher flee reaction to maintain good progress. All techniques other than the *single dog with straight steering* had more difficulty herding the heavy sheep, the seemingly more aggressive technique of the straight steering dog paid off against stubborn sheep.

Interestingly, there was no clear fastest technique for herding a flock with a dominant leader sheep in, presumably because the speed of herding is bounded by how fast the dominant sheep decides to move. Multiple sheepdogs had no trouble herding a flock with outliers, while the single dog, using either technique had problems, taking a significantly longer time to herd the flock. We see a possible explanation for this result in the following test.

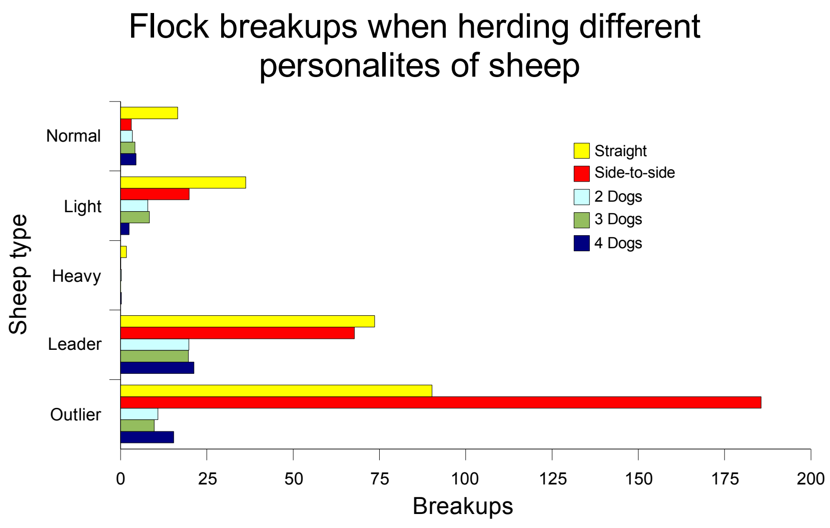


Figure 50: Break ups when herding different personalities of sheep

In general, for any type of sheep, increasing the number of sheepdogs above 2 has a negligible effect on the amount of break ups (fig. 50). This can be explained by the added 'coverage' brought about by more sheepdogs being cancelled out by the potential to disturb the flock when moving around to a new steering point. The addition of a leader sheep proves to be the most disruptive influence on the flock in general.

As with other tests, the outlier sheep has provided strange results, we can assume that the single sheepdog steering techniques were simply not effective enough to control a flock with outlier sheep in. This test again highlights the expected behaviour of the light and heavy sheep, with the light sheep flock being more susceptible to break ups and the heavy sheep flock rarely separating.

8.5 Evaluation of Results

It was shown that the task of gathering sheep can be made more efficient by adding obstacle avoidance, to allow the sheepdog to approach its target without disturbing other sheep. Similarly, using a simple cooperation algorithm, a reasonable speedup can be achieved by adding more sheepdogs to the simulation. Further gathering tests were run using the various sheep personalities that were implemented. It was proved that the dominant sheep made it more difficult to gather the flock by leading other sheep astray. Light sheep were slightly easier to gather due to their increased response to sheepdogs while heavy sheep made the gathering task harder because of their stubbornness. The outlier sheep personality gave mixed results for gathering, making it easier in some cases such as in a flock of heavy sheep and making little difference in others.

The more advanced sheepdog agent worked on flocks rather than single sheep. Although originally intended to be used to herd flocks, it was found that using the flock merging techniques [section 6.5.3], it could also gather sheep. Various algorithms were investigated, showing that working with flocks to gather sheep was more efficient than the single sheep gathering methods.

Following the gathering tests, various tests were run to investigate how well the sheepdog could herd a flock through a series of predefined goals, examining the speed in which this was done as well as the number of flock break ups which occurred. It was shown that varying the steering point for a flock impacted the speed in which it could be herded. Frustration was used as a means of varying the steering point for sheepdogs, this proved to be a suitable replacement for manual steering point modification. Straight steering and side-to-side steering were compared, showing that they both had their advantages. Straight steering was faster but caused more flock break ups, while side-to-side was much slower but much more controlled. In general, the introduction of multiple sheepdogs gave improvements in both respects, showing that using simple cooperation rules, the model allows sheepdogs to work together to achieve a common goal more efficiently.

Finally, we evaluated the effects of emotion and personality when herding sheep using various steering techniques. In the same way as the gathering task, the outlier sheep provided some difficult to explain results, suggesting that the model for this sheep needed more work. The dominant sheep showed interesting results, since it seemed to impose a maximum speed in which any type of sheepdog could herd the flock. The light and heavy sheep worked as expected, the light sheep being easier to herd more quickly but with more break ups and the heavy sheep taking longer but with almost no break ups, regardless of technique. In one case the use of the previously more controlled side-to-side technique actually did more harm than good (fig. 50), showing that there is more work still to be done on advanced steering techniques.

Chapter 9, Conclusion

9.1 General Findings

A model was implemented which successfully simulated the behaviour of sheep and the emergent behaviour exhibited by sheep when in flocks. It was found that a reasonable model could be developed by simply using Reynolds' [1987] original flocking rules, changing the way in which they are combined to tailor the model to the behaviour of sheep. Furthermore, a more realistic looking simulation was achieved by including the work of Brooks [1986], allowing various competing behaviours to be layered into a hierarchy. The task of adding realistic sheep reactions to predators had a very simple solution, by simply adding a highly scaled separation force, focused solely on predators. The addition of this single rule instantly opened up the opportunity to model the behaviour of sheepdogs. Therefore *Research Objectives 1 & 2* were met in a satisfactory manner.

From the extensive work by Lien et al., [2004, 2005], good progress was made into the development of a sheepdog agent, capable of gathering and herding both single sheep and flocks of sheep as a whole (*Research Objective 3*). It was found that some of the ideas from Reynolds' [1987] flocking rules could be used to model facets of the sheepdog's behaviour, in the same way that Barry and Dalrymple-Smith [2005] used attraction and repulsion forces to model the cooperative hunting of lions. However, it was evident that something much more substantial was required to model the various shepherding behaviours. A state-based model was used to split up the sheepdog's various actions between steering, approaching and idling.

Various algorithms were investigated for the decision making aspect of sheepdog behaviour, as well as multiple techniques for steering the flock. The effectiveness of these algorithms and techniques was thoroughly examined through model testing, providing interesting data based on the number of sheep in a simulation and the task at hand. It was found that obstacle avoidance behaviour could be used to emulate 'safe-zone' approaching as discussed by Lien et al., [2004]. This behaviour made the gathering task faster, causing less overall disturbance of the sheep. The most effective method of gathering flocks was to merge flocks using greedy algorithms, merging the two nearest flocks each time until all sheep had been gathered. When herding the flock, side-to-side steering was found to offer more control, causing less flock 'break ups', whereas straight-steering was faster, even though it caused more 'break ups'.

A significant challenge was overcome in developing a subsystem to decide when flocks exist and which sheep are within them. This was very successful as it enabled the project to move forwards from using a sheepdog that simply gathered sheep, to one that would merge and herd flocks efficiently. However, the subsystem was a simplification of the real issue of modelling a sheepdog's perception of the sheep around it and its mental image of where the flock is. Similarly, another simplification was made in the goal setting and goal checking part of gathering and herding. There was no shepherd in the simulation to guide the sheepdogs and decide when the task is complete, therefore, an all knowing entity was responsible for informing the sheepdogs of the goals and their level of completion.

A method of cooperation between sheepdogs was developed, using spontaneous coordination as mentioned by Shibata et al. [1996], the sheepdogs worked together without any communication, the coordination effectively emerged from their local behaviours. Again, some of the original flocking rules were used, such as separation from other sheepdogs in order to spread out with respect to a flock. Some additional decision making was required, the sheepdogs made assumptions about where they believed other dogs may be going and updated their own intentions accordingly. Using relatively simple ideas, the sheepdogs did work together to achieve the same task. The model testing showed that the use of multiple sheepdogs gave a worthwhile speedup and increase in control when both gathering and herding flocks, satisfying *Research Objective 4*.

The ideas of emotion and personality in sheep were investigated, producing some interesting results in testing and providing a good variation with which to test sheepdog interaction. A great deal of progress was made by simply modifying the weightings of the flocking rules based on personality, the additional idea of modifying the neighbourhoods or circles of influence for each flocking rule made it possible to subtly alter the sheep's personality with little extra work.

Two new ideas were added to the sheep model in the form of leadership and fear values. The leadership modifier made it possible for sheep to have different responses to individual neighbours, giving a greater degree of variation than simply varying the overall weighting of each flocking rule. Similarly, the fear value made it possible to model the after effects of being pursued by a predator, causing the flocking desire to persist, as well being able to model the stubbornness to be herded exhibited by some sheep breeds [Downe, 2005]. This allowed the model to include light and heavy sheep, each exhibiting differing degrees of flocking desire and providing different challenges for the sheepdogs. Using the leadership modifier a dominant sheep was modelled which was inclined to lead the flock and potentially stand up to predators. This was proven in testing to make the task of herding much more difficult. Similarly, the light and heavy sheep exhibited distinctive characteristics when interacting with sheepdogs. Personality was used as a means of changing the 'base level' weightings and neighbourhood sizes of the flocking rules. Emotion was used to modify these rules on a per-step basis. The combination of these two ideas showed a good deal of progress into the original aim of this project (Research Objectives 5 & 6).

The project did not manage to completely fulfil all of its original objectives. A relatively small amount of progress was made in terms of investigating different sheepdog personalities and characteristics (Research Objective 7). The use of frustration to help break deadlocks when working with sheep was reasonably successful but could still be much improved At the same time, the simple personality model used for a sheepdog, giving it a specified level of *respect* which affected the sheep's reaction to it, did work as expected, but was only scratching the surface of what could be done. The description of the various sheepdog techniques in the literature review [section 2.2.2], highlights the abundance of potential work still left to do in this area.

9.2 Critical Evaluation of the Model

The fundamental idea of steering another animal using repulsive forces about a steering point was first thought of by Lien et al., [2004]. The model made use of this idea and extended it to include both a steering point and steering pivot. This was a useful improvement as it guaranteed safe-zone approaching as described by Lien et al., [2004]. This method was preferable to the idea proposed by Barry and Dalrymple-Smith [2005], which focused on using cohesive forces between the predator and the prey or in this project's case, the sheepdog and the flock. Although the latter method would have been elegant since it used a variation of Reynolds' [1987] flocking rules, it would have given less flexibility when designing steering behaviours other than straight steering.

The general idea behind animal locomotion was to combine a steering force with the animals movement on the last step (momentum), with a weighted average depending on the type of animal. Generally this worked well, providing realistic movement and also representing the superior agility of dogs over sheep. A decision was made early on to put a constraint on sheep movement to prevent them from straying too close to the edge of the field. This removed the need for developing extra sheepdog behaviours to work on forcing a sheep away from the edge, without being able to get to the steering point in the usual way (since the sheepdog is also bounded by the edges of the field). By preventing the sheep from reaching the edge, the sheepdog was always able to steer them in the normal manner, allowing the project to focus on its research objectives rather than corner cases such as this. Occasionally a strange phenomenon would occur whereby the sheep would become stuck in a line formation, with all of their steering behaviours balanced in one dimension. The addition of a small degree of randomness to the flocking rules broke the deadlock in these cases.

The use of obstacle avoidance in sheepdogs, to avoid existing flocks when approaching the steering point was a very simple but effective addition to the sheepdog's behaviour. In general the obstacle avoidance worked well, although when examining the simulation closely, step-by-step, the sheepdog sometimes changed direction rapidly, looking slightly unnatural. A behaviour which emerged out of the obstacle avoidance and the side-to-side steering technique was that *of arc steering.* When the flock was particularly large, such that the sheepdog was actually going side-to-side *around* the flock, an arc was made to steer the flock. This worked well but was not consistent enough to be classed as a behaviour in its own right.

The way in which the sheep personalities were implemented was very successful. The modifications to the weightings and neighbourhood sizes of the sheep's flocking rules, along with the emotional subsystem, provided a flexible method for modelling a diverse range of sheep personalities. One particular problem was highlighted, the leader sheep would sometimes get 'trapped' against a wall, unable to escape from its own flock members. The flock was strongly attracted to the leader sheep, making it impossible, using flocking rules, for the leader sheep to force its way through the flock, away from the wall.

The testing results showed that sheepdog frustration could increase the speed of herding in some cases, but increased the chances of flock break-ups in others. The model only provided the choice between no emotion and a fixed level of reaction to frustration, making a full investigation into this idea difficult. The frustration mechanism was ineffective when using multiple sheepdogs, since each sheepdog had no idea of the level of frustration any other sheepdog was feeling. The simple hunger mechanism caused sheep to sometimes stand still, rather than wandering or flocking all of the time, while this looked good, it did very little towards simulating the competition between the sheep's desire to flock and graze.

The model of the herding dog was very successful, the way in which it was implemented made it easy to include new techniques and algorithms for deciding how to merge, steer and gather flocks. Comparisons of the gathering sheepdog to the herding sheepdog showed how using this framework made it easy to implement three separate algorithms and make them interchangeable based on the model set up. This allowed the herding dog to be used as a gathering dog, by simply altering the merging algorithm, demonstrating the flexibility of the framework.

The way the 'field of sheep' was implemented worked very well for this project. The fact that it provided a global view for all sheepdogs was somewhat unrealistic, although it did provide an efficient way of simulating sheepdog perception. Representing flocks as a bounding box was simple and efficient but not always effective. Deciding how close to put the steering points to the flock, for any number of dogs, with any steering technique, was a difficult problem. Multiple herding dogs needed a closer steering point to make use of improved control whilst still making progress, however this meant that when they needed to gather sheep, they had problems approaching because of the closer steering point. For some of the testing, the steering point had to be manually modified for different steering techniques in order to get meaningful techniques. Overall we did not find a solution to the problem of working out where to put the steering point for an arbitrarily sized and shaped flock.

9.3 Technical Evaluation

MASON was an exceptionally useful tool for this project. It provided a wealth of features and functionality that otherwise would have needed to have been developed from scratch. This allowed more time to be spent on developing the model and the ideas behind it, rather than focusing on *how* to implement these ideas. Some limitations in the way we used MASON did exist, for example, the model had 5 (including the standard) personalities hard-coded into it, allowing any arbitrary combination of sheep to be used. However, there was no way, with the current system design, to allow the user of the model to tweak the personalities at run-time. We mentioned in the design section that MASON was limited in some ways as it did not make use of Java's generic types. The storage of the agents in the environment relied on static casts when retrieving objects. The use of separate environments for the sheep and sheepdogs alleviated this problem, however it did add a small amount of extra logic at various places in the code, in order to decide which environment to query. The way in which MASON provided a model parameters page at runtime was very convenient, however it required many *getter* and *setter* methods to be added to the simulation class. This was more a question of style, as it did make the main simulation class large and cluttered.

The use of Java, and the decisions made when designing the class hierarchy made a well structured, extendable and reusable system. The code-base could be extended in the future with minimal effort to create new types of sheep or sheepdog, being able to seamlessly reuse any techniques and algorithms already implemented.

As development of the sheepdog agents progressed, it became clear that the state-based approach could have been implemented better if it had been designed from the outset. As it stands, the sheepdog agent stores a value representing the state which it is in and checks it many times to decide which action to perform. A possible alternative to this, would have been to use the *state design pattern* [Gamma et al., 1994]. This would allow interchangeable state objects to be used within the sheepdog class, removing the potential for many large and unwieldily switch statements.

Unit testing was partly used, to test the mathematical side of the model such as the functions written for performing operations and comparisons on vectors, regions and distances. This provided a certain level of confidence in the utility functions used by the model. In terms of testing the actual agents themselves, little progress was made. Many of the desired results came about through emergent behaviour, which is difficult to both design and test for [Wooldridge, 2002]. In the future, mock objects and mock environments could be designed to allow a basic level of testing for the agents and their interactions. However, unit testing of agent-based models is still a relatively new area of research, and there is no clear best solution [Coelho et al., 2006].

9.4 Future Work

The following section identifies potential improvements and extensions to the project. We split this section into four parts: future work on modelling sheep, on modelling sheepdogs, general future work, and further work or research relating to the animals in question.

9.4.1 Future Work on Modelling Sheep

Sheep are known to communicate fear through pheromones [Downe, 2005][Delgado-Mata and Aylett, 2004]. This could be incorporated into the model with minimal effort due to the simple layered hierarchy used. When updating the fear emotion, a sheep would check its neighbours to see how much fear they were feeling, and then update their own fear level appropriately. The amount of attention paid to other sheep's fear level could be modelled as a function of the sheep's personality, for example, sheep would pay more attention to the fear level of a leader sheep. This could change the way a flock flees from a predator, since the entire flock would begin to move, even before the sheepdog has reached the flight zone of each sheep. Reynolds [2000] implemented something similar to this, in which pigeons transition between *walking* and *flying* depending on their neighbours' states. Reynolds referred to this idea as *state contagiousness*.

Another interesting idea, presented by Reynolds [2000] was that of a neighbourhood which elongates based on the animals current speed. Currently we model the animals neighbourhood as a circle. However, it could be interesting to make the animals look further ahead in the direction they are travelling. This could go some way to simulating the bounded vision of predators, rather than the 360 degree vision used in this project. Also the consideration of a sheepdog's speed of approach could be considered in the sheep's flee and fear reaction, with a quickly approaching sheepdog putting the flock into a state of panic. Without this idea, it was feasible for a sheepdog to reach the sheep before it has even started moving, when comparing this with real predator-prey hunting, it is unrealistic.

The Brooks [1986] layered subsumption architecture was used with great success in the sheep model. A further investigation could be performed to see if we could move away from the traditional combination of flocking rules, representing fleeing from a predator as a separate layer, or even further, representing each flocking rule as a separate layer.

It has been observed that related sheep are more likely to form sub-groups within a flock, as are same-breed sheep in a mixed-breed flock, as well as a ewe and its descendants [Weaver, 2005]. The ideas of these sub-flocks could provide an interesting extension to the model, giving the sheepdog more to consider when trying to round up a flock. An idea touched on by Downe [2005] highlights how sheep alter their desire to perform certain flocking rules as their hunger changes. This would be reasonably simple to implement using the current model, either by adding a new layer to the hierarchy which uses separate flocking rules, or by modifying the flocking rules to always take into account the current hunger level.

9.4.2 Future Work on Modelling Sheepdogs

There is obviously a great amount of work to be done towards providing a more realistic simulation of dog vision in a 2D environment. This would involve taking into account the forward facing vision of a dog, calculating which sheep can actually be seen, taking into account the issue of not being able to see 'through' sheep, and therefore not having an entire view of a flock. The obstacle avoidance behaviour used in the sheepdog was a form of forward facing vision, since it only considered sheep that were straight in front of the sheepdog. Taking this into account, perhaps a slightly more realistic sheepdog vision was not so far away. However, with limited vision, an algorithm or strategy would need to be developed for the sheepdog to be able to search the field for flocks.

As already discussed, side-to-side steering was an effective method of controlling the flock while steering, however it did not always work as expected and some strange results were observed. Further work could be done into making this behaviour more effective, such as deciding where to place the steering points, how wide to make each left/right alternation and how close to get to the flock when moving side-to-side. We observed arc steering as an emergent behaviour of the sheepdog, this could be investigated further, as a more concrete steering technique, to see if it was more consistent than side-to-side steering.

As mentioned in the critical evaluation, there was no flexibility in the way frustration was modelled in sheepdogs, another personality trait could be added to the sheepdog personality, such as *patience,* representing how easily the sheepdog will become frustrated. An improvement could also be made to the mechanism used for updating sheepdog frustration. Currently frustration is increased when the sheepdog is moving at a particular fraction of its maximum speed, for straight steering this worked reasonably well, however in side-to-side steering, where the sheepdog is always moving at its maximum speed, it was useless. The measure of frustration could instead be related to the speed of the flock which is being driven, rather than the sheepdog itself, since this relates more directly to the success of the task and less directly to *how* the task is being worked on. Additionally, the size of the side-to-side movements could be influenced by some form of emotion, based on how compact the flock is, allowing the sheepdog to vary its steering technique depending on how much control it feels the flock is under.

A fundamental change to the simulation could be the way in which the sheepdog steers the flock. Currently the sheepdog is always actively driving the flock towards its goal. Instead, it would be possible for the sheepdog to push the flock in a particular direction and then let its momentum carry it towards the goal, only acting on the flock if it strays away from the desired direction. The framework for this is already in place in the project, since the system provides information on the average speed of a flock, allowing sheepdogs to predict where the flock is heading. Performing herding in this way would reduce the amount of time in which the sheepdog is in the flocks flight zone, potentially reducing stress while herding. Similarly, the sheepdog could focus on the most dominant sheep in the flock, and herd this, causing the rest of the flock to follow without needing to directly influence them.

The style of hunting used by lions was investigated in the literature review and some of the cooperation mechanisms were used as inspiration for multiple sheepdog cooperation. A more explicit method of this idea could be used by multiple sheepdogs to identify particularly difficult sheep such as the *outlier sheep*, whereby the sheepdogs surround it closely, as if hunting it.

The way in which safe-zone approaching was implemented relied on obstacle avoidance. In the future this could be implemented to only take account of what is counted as a flock (an arbitrarily sized group of sheep). This would allow the sheepdog to avoid breaking up large flocks whilst being able to make better progress by not performing time-consuming obstacle avoidance around every single sheep in the field.

9.4.3 General Future Work

To add more realism and variety to the environment, arbitrary obstacles could be added to the environment. This would extend the sheepdog's obstacle avoidance behaviour used for avoiding sheep whilst approaching the flock. A similar situation would have to be added to the sheep model to allow the flock to steer round obstacles, without getting trapped up against them. Moreover, the sheepdog may also need to take in to account some path finding algorithms when herding sheep to a goal, so that it takes the flock down a route that will ultimately end up at the goal, rather than a dead end [Lien et al., 2004].

A more creative addition to the model would be the ability for users to manually control the sheepdog. This would give the user freedom to explore sheep and sheepdog interactions in any number of ways, and would perhaps give a more useful insight into how the different personalities of sheep react. Furthermore, with a realistic sheep flocking model implemented, the development of a sheepdog agent could be used as a way of teaching a practical approach to an agents course or as a fun agents competition.

The use of the 'oracle' for assigning and assessing goals made it easy for multiple sheepdogs to work together. This feature may make it possible for a shepherd to be added to the model in the future, where the shepherd can make decisions and issue separate tasks to sheepdogs to accomplish some overall task.

9.4.4 Future Animal Research

As well as the computer science oriented ideas above, there are some ideas that would require further background research into sheep, sheepdogs and the dynamics between the two. It would be interesting to further investigate how fear effects the behaviour of sheep both during and after a predator has been pursuing them. For example, does the number of sheep and the dominance of surrounding sheep alter the speed in which a sheep's level of fear fluctuates?

In the model, the sheepdogs did not make any differentiation between a normal, light or heavy sheep, they simply applied the same technique to each sheep, with varying success, as shown in the model results (Chapter 8). An intriguing piece of research would be to assess whether or not sheepdogs actually exhibit some form of recognition of 'difficult' sheep, and change their approach accordingly.

As mentioned earlier, more realistic sheepdog vision would be worthwhile future work. This would rely on finding more material describing how dogs perceive their environment, for example, do they focus on moving objects more prominently? As well as understanding how a sheepdog can remember its environment, in order to build a mental image of the shape of the field and the position of sheep within it.

The evaluation of steering techniques took into account flock break ups, the significance of this would require more research into the effects this can have on the flock, such as increased panic, fatigue or other behavioural modifications.

9.5 Summary

Overall, the project has thoroughly explored all but one of the original research objectives. We have shown how a successful sheep flocking model was developed, using ideas from Reynolds [1987], Brooks [1986] and original ideas based on emotion and personality of sheep. A working model of a sheepdog was created, able to perform tasks such as gathering and herding sheep. We have demonstrated a method in which multiple sheepdogs can cooperate, without any external coordination such as a shepherd, showing that the time taken to achieve a task can be reduced by adding more sheepdogs. Similarly we have investigated two main single sheepdog steering techniques, showing how even with a single sheepdog, an improved level of control can be reached by using side-to-side steering. A method for identifying flocks was developed which made it possible to develop a generic algorithm for gathering, herding and merging arbitrarily sized flocks.

The model has demonstrated a method of simulating different personalities of sheep. The task of the sheepdog is shown to be made more difficult with the addition of a dominant sheep, since the dominant sheep is seen to lead the flock away from its intended location. We have identified many possible areas of future work both in terms of modelling sheep and sheepdogs as well as looking at more general predator-prey interactions. The work done so far may be useful for creating realistic shepherding behaviours for use in robotics and even in the film and computer game industries.

Bibliography

Adachi, Y., Kakikura, M., (2006). *Research on Sheepdog Problem Using Cellular Automata.* Computational Cybernetics, 2006 IEEE International Conference on.

Barry, A. M., Dalrymple-Smith, H. (2005) *Visual Communication and Social Structure - The Group Predation of Lions.* Technical report, Department of Computer Science, University of Bath, UK, 2005.

BBC News Website. (2004). *Crafty sheep conquer cattle grids.* Available from: http://news.bbc.co.uk/1/hi/uk/3938591.stm [Accessed 19/04/08]

Bonabeau, E. (2002) *Agent-based modeling: Methods and techniques for simulating human systems.* PNAS 2002; 99: 7280-7287.

Brooks, R. A., (1986). *A Robust Layered Control System For A Mobile Robot.* IEEE Journal of Robotics and Automation, Vol. RA-2, No.I, March 1986.

Budiansky, S. (1999). *The Covenant of the Wild: Why animals chose domestication.* Yale University Press.

Cami, A., Lisetti, C., Maarten, S. (2003). *Accounting for Emotions in Multi-Agent Modeling and Simulation Systems.* Proceedings UM'2003, 9th International Conference on User Model, June 22-26, 2003, Pittsburg, USA.

Coelho, R., Kulesza, U., von Staa, A., Lucena, C. (2006) *Unit Testing in Multi-Agent Systems using Mock Agents and Aspects.* Computer Science Department, Pontifical Catholic University of Rio de Janeiro.

De Vos, M., Padget, J. (2007). *Agents and Electronic Commerce: Agent-Based Modelling*, CM30174 Lecture Notes, 2007.

Delgado-Mata, C., Aylett, R. S. (2004). *Emotion and Action Selection: Regulating the Collective Behaviour of Agents in Virtual Environments.* AAMAS'04, July 19-23, 2004, New York, New York, USA.

Delgado-Mata, C., Ibanez-Martinez, J., (2006). *An Emotion Affected Action Selection Mechanism for Multiple Virtual Agents.* Proceedings of the 16th International Conference on

Artificial Reality and Telexistence--Workshops (ICAT'06).

Dictionary.com (2008). *Dictionary.com definition "Artificial Life".* Available from: http://dictionary.reference.com/browse/artificial%20life [Accessed 20/04/2008]

Downe, N. J. (2005) *Refinement of Reynolds' flocking rules in a distributed behavioural model to better reflect the semi-adversarial interaction between sheep and sheepdogs.* Final Year Project, University of Bath.

Fight or Flight. In: *Wikipedia: the free encyclopaedia [online].* St Petersburg, Florida: Wikimedia Foundation. Available from: http://en.wikipedia.org/wiki/Fight_or_flight [Accessed 13/02/2008]

Flight Zone. In: *Wikipedia: the free encyclopaedia [online].* St Petersburg, Florida: Wikimedia Foundation. Available from: http://en.wikipedia.org/wiki/Flight_Zone [Accessed 13/02/2008]

Gabbai, J.M.E., (2005) *Complexity and the Aerospace Industry: Understanding Emergence by Relating Structure to Performance using Multi- Agent Systems.* PhD Thesis, Faculty of Engineering and Physical Sciences, The University of Manchester.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1994) *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Professional Computing Series

Gardner, M., (1970) *Mathematical Games - The fantastic combinations of John Conway's new solitaire game "life".* Scientific American 223 (October 1970): 120-123.

Gueron, S., Levin, S.A., Rubenstien, D.I., (1996). *The Dynamics of Herds: From Individuals to Aggregations.* J . theor . Biol . (1996) 182, 85–98.

Gill, W. (2004) *Applied Sheep Behaviour.* Animal Science Department, The University of Tennessee.

Gulyás, L., (2005). *Understanding Emergent Social Phenomena.* Hungarian Academy of Sciences, 2005.

Hartigan, J.A. (1975) *Clustering Algorithms.* New York: John Wiley & Sons, Inc.

Herding Dog. In: *Wikipedia: the free encyclopaedia [online].* St Petersburg, Florida: Wikimedia Foundation. Available from: http://en.wikipedia.org/wiki/Herding_Dog [Accessed 13/02/2008]

Joubert, D., (2006). *Hunting behaviour of lions (Panthera leo) on elephants (Loxodonta africana) in the Chobe National Park, Botswana.* African Journal of Ecology 44 (2) , 279–281.

Leventhal, H., Scherer, K. (1987) *The relationship of emotion to cognition: A functional approach to a semantic controversy.* Cognition and Emotion, vol. 1, no. 1, pp. 3–28, 1987.

Li, T., Ma, Y., Qiu, Y., Peng, Y., (2007). *Modelling Personality, Emotion and Mood for a Pedagogical Agent.* Proceedings of the 25th IASTED International Multi-Conference, Artificial Intelligence and Applications, February 12-14 2007, Innsbruck, Austria.

Lien, J., Bayazit, O. B., Sowell, R.T., Rodriguez, S., Amato, N.M., (2004). *Shepherding Behaviors.* Proceedings of the 2004 IEEE International Conference on Robotics and Automation, New Orleans, LA, April 2004, pp 4159-4164.

Lien, J., Rodriguez, S., Malric, J., Amato, N. M., (2005). *Shepherding Behaviors with Multiple Shepherds.* Proceedings of the 2005 IEEE, International Conference on Robotics and Automation, Barcelona, Spain, April 2005, pp 3402-3407.

Lindhé, M. (2004) *A Flocking and Obstacle Avoidance Algorithm for Mobile Robots.* Master's Degree Project, Stockholm University, Sweden.

Luke, S., Sullivan, L., Panait, L., Balan, G. (2005) *Tunably Decentralized Algorithms for Cooperative Target Observation.* Department of Computer Science, George Mason University.

MASON Multiagent Simulation Toolkit. (2008) http://www.cs.gmu.edu/~eclab/projects/mason/ [Accessed 09/03/2008]

McFarland, D., (2006). *A Dictionary of Animal Behaviour.* Oxford University Press, 2006.

McNutt, J., Malcolm, J., (2007). *"African Wild Dog" The Encyclopaedia of Mammals.* Ed. David W. Macdonald. Oxford University Press, 2007. Oxford Reference Online. Oxford University Press. Available from: http://www.oxfordreference.com/views/ENTRY.html?subview=Main&entry=t227.e [Accessed 17/02/2008]

Matsumoto, M., Nishimura, T. (1998) *Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator.* ACM Transactions on Modeling and Computer Simulation (TOMACS). Volume 8 , Issue 1 (January 1998). p3-30.

Murphy, R. R., Lisetti, C. L., Tardif, R., Irish, L., Gage, A. (2002). *Emotion-Based Control of Cooperating Heterogeneous Mobile Robots.* IEEE transactions on robotics and automation, vol. 18, no. 5, October 2002, pp.744-757.

Nickless, A., Watson, G., (2008). *Working Sheepdog Website : Basic Techniques and Information for Training Border Collie Sheepdogs* [online]. Available from: http://www.workingsheepdog.co.uk/basic-sheepdog-training.htm [Accessed 13/02/2008]

Ortony, A., Clore, G.L., Collins, A., (1998). *The Cognitive Structure of Emotions.* Cambridge University Press, 1988.

Paquet, P. C., Zimen, E., Boitani, L., (2007). *"Wolves" The Encyclopedia of Mammals.* Ed. David W. Macdonald. Oxford University Press, 2007. Oxford Reference Online. Oxford University Press. Available from: http://www.oxfordreference.com/views/ENTRY.html?subview=Main&entry=t227.e160

[Accessed 17/02/2008]

Parunak, H., Savit, R., Riolo, R. L. (1998), *Agent-Based Modelling vs. Equation Based Modelling: A Case Study and Users Guide.* Proceedings of Multi-agent systems and Agent-based Simulation (MABS'98), 10-25, Springer, LNAI 1534, 1998.

Potts, W.K., (1984). *The chorus-line hypothesis of manoeuvre coordination in avian flocks.* Nature 309, 344 - 345 (24 May 1984).

Prey Drive. In: *Wikipedia: the free encyclopaedia [online].* St Petersburg, Florida: Wikimedia Foundation. Available from: http://en.wikipedia.org/wiki/Prey_drive [Accessed 13/02/2008]

Pocknell, P., (1999). *An Investigation Into Computational Flocking Techniques* [online]. Final Year Dissertation (Bsc.) Middlesex University. Available from: http://www.citizenphil.co.uk/flocking.html [Accessed 17/02/2008]

Railsback, S. F., Lytinen, S. L., Jackson, S. K., (2006) *Agent-based Simulation Platforms: Review and Development Recommendations*. Simulation 82: 609-623.

Reynolds, C. W., (1987). *Flocks, Herds, and Schools: A Distributed Behavioral Model.* Computer Graphics, Volume 21, Number 4, July 1987.

Reynolds, C. W., (1999). *Steering Behaviors For Autonomous Characters, in the proceedings of Game Developers Conference* 1999 held in San Jose, California. Miller Freeman Game Group, San Francisco, California. Pages 763-782.

Reynolds, C. W., (2000). *Interaction with Groups of Autonomous Characters*, in the proceedings of Game Developers Conference 2000, CMP Game Media Group (formerly: Miller Freeman Game Group), San Francisco, California, pages 449-460.

Schaller, G, B., (1976). *The Serengeti Lion: A Study of Predator-Prey Relations.* University of Chicago Press.

Sheep, Animal Corner. (2008). *Sheep at Animal Corner* [online]. Available from: http://www.animalcorner.co.uk/farm/sheep/sheep_about.html [Accessed 13/02/2008]

Shibata, T., Ohkawa, K., Tanie, K., (1996). *Spontaneous Behavior of Robots for Cooperation - Emotionally Intelligent Robot System.* Proceedings of the 1996 IEEE International Conference on Robotics and Automation.

Simmons, P., Ekarius, C., (2001). *Storey's Guide to Raising Sheep.* North Adams, MA: Storey Publishing LLC

Smith, B.,, Aseltine, M., Kennedy, G. (1997). *Beginning Shepherd's Manual, Second Edition.* Ames, Iowa: Iowa State University Press.

TSP. (2008). *The Travelling Salesman Problem [online].* Georgia Institute of Technology. Available from: http://www.tsp.gatech.edu/problem/index.html [Accessed 20/04/2008]

Vaughan, R., Sumpter, N., Henderson, J., Frost, A., Cameron, S. (1998) *Experiments in automatic flock control.* Proceedings of the 1998 IEEE ISIC/CIRA/ISAS Joint Conference Gaithersburg, MO 0 September 14-17, 1998. pp 277-282.

Weaver, S. (2005). *Sheep: small-scale sheep keeping for pleasure and profit.* 3 Burroughs Irvine, CA 92618: Hobby Farm Press, an imprint of BowTie Press, a division of BowTie Inc.

Wiggins, J. (1996). *The Five Factor Model of Personality: Theoretical Perspectives.* New York, The Guilford Press, 1996.

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. John Wiley & Sons Ltd.

Working Dogs, Encyclopaedia of New Zealand, (1966). *Working Dogs. In: Encyclopaedia of New Zealand,* edited by A. H. McLintock, originally published in 1966. Available from: http://www.teara.govt.nz/1966/D/DogsWorking/DogsWorking/en [Accessed 13/02/2008]

Appendix A - Technology Overview

MASON

MASON provides an extensive array of library classes to aid in the development of a multi-agent simulation. A basic graphical interface is also provided, this meant that this project could focus on implementing the behaviours of the agents, rather than having to work on the underlying framework.

Inspectors

A useful feature, both for the development of the project and the results gathering stage is that of the inspector. MASON allows agents to be inspected at runtime, this involves allowing the user to view any public fields or public getter methods, as well as being able to change the agents state at runtime through any public setter methods. Global parameters for the model can also be changed, allowing different scenarios to be tested without having to run separate programs.

Agents

Agents are represented the *the Steppable* interface. Each class that implements the S*teppable* interface will be capable of receiving an update based on the current state, on every step of the simulation.

Steps

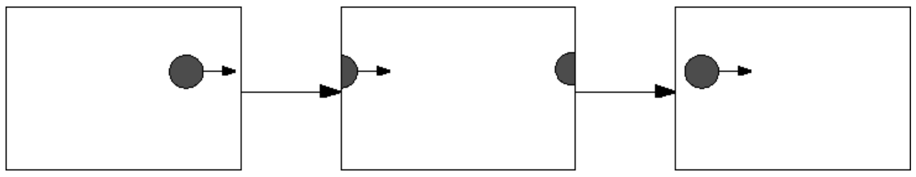
A single unit of time in the simulation is known as a step. On each step, all objects implementing the Steppable interface receive a *SimState* object, providing information about the current state of the simulation.

2D Environment

A continuous environment was used, rather than a discrete environment. The continuous environment provided more flexibility when it came to the movement of animals, both in terms of direction and magnitude.

MASON provides a *Continuous2D* class which allows agents to be stored in a hashmap using an (x,y) coordinate. It was important to consider the practical implications of using this type of environment, MASON stores objects in a continuous 2D environment by maintaining a series of buckets and by performing hash look-ups when searching for objects in a particular location. Because of this, calls to the environment to look for an object or to find objects within a specific range are expensive. Therefore, the location of an agent was also stored within the agent itself, the disadvantage of having redundant data was outweighed by the increased efficiency.

MASON also provides the ability to have a *toroidal* environment, that is, the environment wraps round at the edges, as demonstrated below:



During the early prototyping of the model, this feature was useful, because it removed the need to consider how animals react to reaching the edge of the field. When implementing the full model, this feature was not used, the environment was bounded and the animals avoided walls appropriately.

It is possible to store objects of different types in a single Continuous2D environment, however, the loss of type information when retrieving objects from the environment makes this impractical. To overcome this limitation, several Continuous2D environments were used. The logical distinction for this model was between predators and prey. The logic relating to how to combine the information from both of these environments was left up to the agent.

Bags

The main storage mechanism used by MASON is a *Bag*. A Bag is a similar collection to that of an array list, without the use of generic types that were introduced in Java 1.5. Although the use of generic types would have been useful, bags are said to be a faster alternative. Due to the number of operations that are performed over collections of agents, it made sense to accept the trade off and use bags, performing explicit casts where required.

Portrayals

The graphical side of MASON is viewed as a collection of portrayals. Each agent can have an associated portrayal. MASON provides several base classes to use as portrayals, allowing the user to extend their functionality to fit the particular simulation.

The overall SimState as described above is encapsulated by a GUIState, this allows the simulation to be represented graphically without interfering with the underlying model.

Java 2D

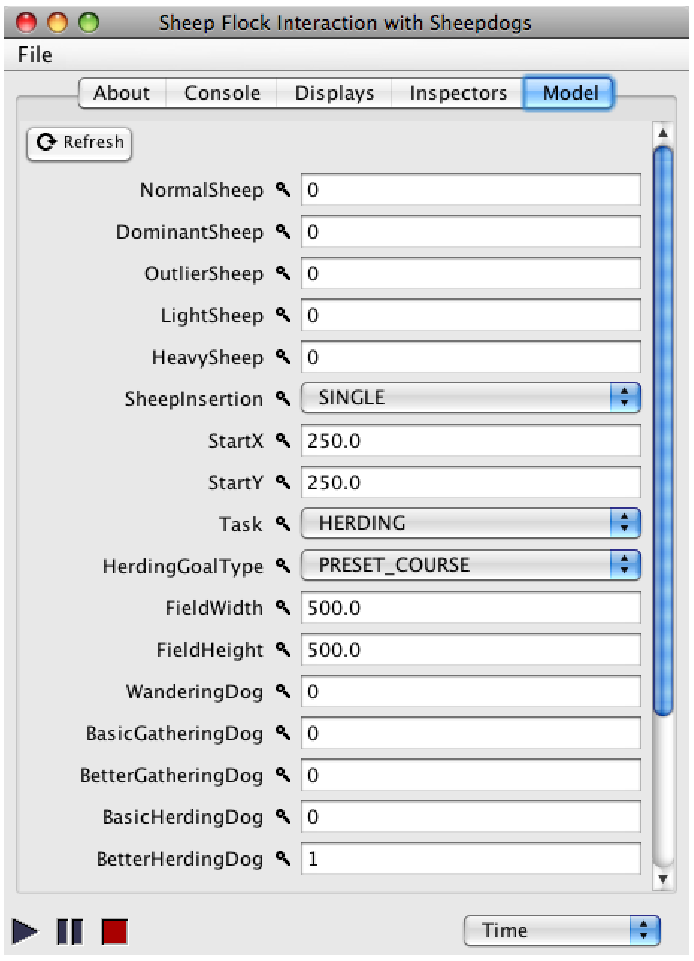
Java provides a comprehensive range of libraries aimed at performing geometric calculations in 2D. Some examples of this functionality was:

- Intersections – The ability to draw a box then see if a point is in it.
- Lines – Checking the distance of a point to a line.
- Bounding boxes – Checking overlap/collision/membership of a bounding box.

The use of these Java2D libraries allowed development to focus on development of the model, rather than the underlying geometry behind it.

Appendix B – Using the Model

The model was run as a graphical interface using MASON. The following screen shot shows the model page of the simulation:



NormalSheep	0
DominantSheep	0
OutlierSheep	0
LightSheep	0
HeavySheep	0
SheepInsertion	SINGLE
StartX	250.0
StartY	250.0
Task	HERDING
HerdingGoalType	PRESET_COURSE
FieldWidth	500.0
FieldHeight	500.0
WanderingDog	0
BasicGatheringDog	0
BetterGatheringDog	0
BasicHerdingDog	0
BetterHerdingDog	1

The model is started, stopped and paused using the buttons in the lower left corner.

The parameters available on the model page are used for running many different configurations of simulation. The following table outlines the key parameters:

Parameter	**Description**
NormalSheep	Number of standard sheep to add to the simulation.
DominantSheep	Number of dominant sheep to add to the simulation.
OutlierSheep	Number of outlier sheep to add to the simulation.
LightSheep	Number of light sheep to add to the simulation.
HeavySheep	Number of heavy sheep to add to the simulation.
SheepInsertion	Single or random. If single, sheep are inserted in one flock.
StartX	If using Single insertion, coordinate of insertion point.
StartY	If using Single insertion, coordinate of insertion point.
Task	Herding or Gathering the flock.
HerdingGoalType	If herding, use a predefined set of goals, or use a continuous random set of goals.
FieldWidth	Width of environment
FieldHeight	Height of environment
WanderingDog	Number of wandering dogs, randomly moving predator.
BasicGatheringDog	Gathering dog with no obstacle avoidance.
BetterGatheringDog	Gathering dog with obstacle avoidance.
BasicHerdingDog	Herding dog with no flock recognition
BetterHerdingDog	Herding dog with full flock recognition (Best Dog)
UseSideToSide	Use the side to side steering behaviour
SheepdogEmotion	Use frustration as a way of modifying steering points.
DebugMode	Show debug information.
HerdingDog SteeringPoint	Steering point distance for a herding dog, can only be changed before simulation is run.
HerdingDog SteeringPivot	Steering pivot distance for a herding dog, can only be changed before simulation is run.